



IrDA-like printer

((L2CAP, PSM=RFCOMM), (RFCOMM, CN=1), (PostscriptStream))

IP Network Printing

((L2CAP, PSM=RFCOMM), (RFCOMM, CN=2), (PPP), (IP), (TCP),
(IPP))

Synchronization Protocol Descriptor Example

((L2CAP, PSM=0x1001), (RFCOMM, CN=1), (Obex), (vCal))

((L2CAP, PSM=0x1002), (RFCOMM, CN=1), (Obex),
(otherSynchronisationApplication))

5.1.6 BrowseGroupList Attribute

Attribute Name	Attribute ID	Attribute Value Type
BrowseGroupList	0x0005	Data Element Sequence

Description:

The BrowseGroupList attribute consists of a data element sequence in which each element is a UUID that represents a browse group to which the service record belongs. The top-level browse group ID, called PublicBrowseRoot and representing the root of the browsing hierarchy, has the value 00001002-0000-1000-8000-00805F9B34FB (UUID16: 0x1002) from the Bluetooth Assigned Numbers document.

5.1.7 LanguageBaseAttributeIDList Attribute

Attribute Name	Attribute ID	Attribute Value Type
LanguageBaseAttributeIDList	0x0006	Data Element Sequence

Description:

In order to support human-readable attributes for multiple natural languages in a single service record, a base attribute ID is assigned for each of the natural languages used in a service record. The human-readable universal attributes are then defined with an attribute ID offset from each of these base values, rather than with an absolute attribute ID.

The LanguageBaseAttributeIDList attribute is a list in which each member contains a language identifier, a character encoding identifier, and a base attribute



ID for each of the natural languages used in the service record. The LanguageBaseAttributeIDList attribute consists of a data element sequence in which each element is a 16-bit unsigned integer. The elements are grouped as triplets (threes).

The first element of each triplet contains an identifier representing the natural language. The language is encoded according to ISO 639:1988 (E/F): "Code for the representation of names of languages".

The second element of each triplet contains an identifier that specifies a character encoding used for the language. Values for character encoding can be found in IANA's database¹, and have the values that are referred to as MIBEnum values. The recommended character encoding is UTF-8.

The third element of each triplet contains an attribute ID that serves as the base attribute ID for the natural language in the service record. Different service records within a server may use different base attribute ID values for the same language.

To facilitate the retrieval of human-readable universal attributes in a principal language, the base attribute ID value for the primary language supported by a service record must be 0x0100. Also, if a LanguageBaseAttributeIDList attribute is contained in a service record, the base attribute ID value contained in its first element must be 0x0100.

5.1.8 ServiceInfoTimeToLive Attribute

Attribute Name	Attribute ID	Attribute Value Type
ServiceInfoTimeToLive	0x0007	32-bit unsigned integer

Description:

The ServiceTimeToLive attribute is a 32-bit integer that contains the number of seconds for which the information in a service record is expected to remain valid and unchanged. This time interval is measured from the time that the attribute value is retrieved from the SDP server. This value does not imply a guarantee that the service record will remain available or unchanged. It is simply a hint that a client may use to determine a suitable polling interval to re-validate the service record contents.

1. See <http://www.isi.edu/in-notes/iana/assignments/character-sets>



5.1.9 ServiceAvailability Attribute

Attribute Name	Attribute ID	Attribute Value Type
ServiceAvailability	0x0008	8-bit unsigned integer

Description:

The ServiceAvailability attribute is an 8-bit unsigned integer that represents the relative ability of the service to accept additional clients. A value of 0xFF indicates that the service is not currently in use and is thus fully available, while a value of 0x00 means that the service is not accepting new clients. For services that support multiple simultaneous clients, intermediate values indicate the relative availability of the service on a linear scale.

For example, a service that can accept up to 3 clients should provide ServiceAvailability values of 0xFF, 0xAA, 0x55, and 0x00 when 0, 1, 2, and 3 clients, respectively, are utilizing the service. The value 0xAA is approximately $(2/3) * 0xFF$ and represents 2/3 availability, while the value 0x55 is approximately $(1/3) * 0xFF$ and represents 1/3 availability. Note that the availability value may be approximated as

$$(1 - (\text{current\_number\_of\_clients} / \text{maximum\_number\_of\_clients})) * 0xFF$$

When the maximum number of clients is large, this formula must be modified to ensure that ServiceAvailability values of 0x00 and 0xFF are reserved for their defined meanings of unavailability and full availability, respectively.

Note that the maximum number of clients a service can support may vary according to the resources utilized by the service's current clients.

A non-zero value for ServiceAvailability does not guarantee that the service will be available for use. It should be treated as a hint or an approximation of availability status.

5.1.10 BluetoothProfileDescriptorList Attribute

Attribute Name	Attribute ID	Attribute Value Type
BluetoothProfileDescriptorList	0x0009	Data Element Sequence

Description:

The BluetoothProfileDescriptorList attribute consists of a data element sequence in which each element is a profile descriptor that contains information about a Bluetooth profile to which the service represented by this service record conforms. Each profile descriptor is a data element sequence whose first element is the UUID assigned to the profile and whose second element is a 16-bit profile version number.



Each version of a profile is assigned a 16-bit unsigned integer profile version number, which consists of two 8-bit fields. The higher-order 8 bits contain the major version number field and the lower-order 8 bits contain the minor version number field. The initial version of each profile has a major version of 1 and a minor version of 0. When upward compatible changes are made to the profile, the minor version number will be incremented. If incompatible changes are made to the profile, the major version number will be incremented.

5.1.11 DocumentationURL Attribute

Attribute Name	Attribute ID	Attribute Value Type
DocumentationURL	0x000A	URL

Description:

This attribute is a URL which points to documentation on the service described by a service record.

5.1.12 ClientExecutableURL Attribute

Attribute Name	Attribute ID	Attribute Value Type
ClientExecutableURL	0x000B	URL

Description:

This attribute contains a URL that refers to the location of an application that may be used to utilize the service described by the service record. Since different operating environments require different executable formats, a mechanism has been defined to allow this single attribute to be used to locate an executable that is appropriate for the client device's operating environment. In the attribute value URL, the first byte with the value 0x2A (ASCII character '*') is to be replaced by the client application with a string representing the desired operating environment before the URL is to be used.

The list of standardized strings representing operating environments is contained in the Bluetooth Assigned Numbers document.

For example, assume that the value of the ClientExecutableURL attribute is `http://my.fake/public/*/client.exe`. On a device capable of executing SH3 WindowsCE files, this URL would be changed to `http://my.fake/public/sh3-microsoft-wince/client.exe`. On a device capable of executing Windows 98 binaries, this URL would be changed to `http://my.fake/public/i86-microsoft-win98/client.exe`.



5.1.13 IconURL Attribute

Attribute Name	Attribute ID	Attribute Value Type
IconURL	0x000C	URL

Description:

This attribute contains a URL that refers to the location of an icon that may be used to represent the service described by the service record. Since different hardware devices require different icon formats, a mechanism has been defined to allow this single attribute to be used to locate an icon that is appropriate for the client device. In the attribute value URL, the first byte with the value 0x2A (ASCII character ‘*’) is to be replaced by the client application with a string representing the desired icon format before the URL is to be used.

The list of standardized strings representing icon formats is contained in the Bluetooth Assigned Numbers document.

For example, assume that the value of the IconURL attribute is `http://my.fake/public/icons/*`. On a device that prefers 24 x 24 icons with 256 colors, this URL would be changed to `http://my.fake/public/icons/24x24x8.png`. On a device that prefers 10 x 10 monochrome icons, this URL would be changed to `http://my.fake/public/icons/10x10x1.png`.

5.1.14 ServiceName Attribute

Attribute Name	Attribute ID Offset	Attribute Value Type
ServiceName	0x0000	String

Description:

The ServiceName attribute is a string containing the name of the service represented by a service record. It should be brief and suitable for display with an icon representing the service. The offset 0x0000 must be added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.



5.1.15 ServiceDescription Attribute

Attribute Name	Attribute ID Offset	Attribute Value Type
ServiceDescription	0x0001	String

Description:

This attribute is a string containing a brief description of the service. It should be less than 200 characters in length. The offset 0x0001 must be added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.

5.1.16 ProviderName Attribute

Attribute Name	Attribute ID Offset	Attribute Value Type
ProviderName	0x0002	String

Description:

This attribute is a string containing the name of the person or organization providing the service. The offset 0x0002 must be added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.

5.1.17 Reserved Universal Attribute IDs

Attribute IDs in the range of 0x000D-0x01FF are reserved.

5.2 SERVICEDISCOVERYSERVER SERVICE CLASS
ATTRIBUTE DEFINITIONS

This service class describes service records that contain attributes of service discovery server itself. The attributes listed in this section are only valid if the ServiceClassIDList attribute contains the ServiceDiscoveryServerServiceClassID. Note that all of the universal attributes may be included in service records of the ServiceDiscoveryServer class.

5.2.1 ServiceRecordHandle Attribute

Described in the universal attribute definition for ServiceRecordHandle.

Value

A 32-bit integer with the value 0x000000000.

5.2.2 ServiceClassIDList Attribute

Described in the universal attribute definition for ServiceClassIDList.

Value

A UUID representing the ServiceDiscoveryServerServiceClassID.

5.2.3 VersionNumberList Attribute

Attribute Name	Attribute ID	Attribute Value Type
VersionNumberList	0x0200	Data Element Sequence

Description:

The VersionNumberList is a data element sequence in which each element of the sequence is a version number supported by the SDP server.

A version number is a 16-bit unsigned integer consisting of two fields. The higher-order 8 bits contain the major version number field and the low-order 8 bits contain the minor version number field. The initial version of SDP has a major version of 1 and a minor version of 0. When upward compatible changes are made to the protocol, the minor version number will be incremented. If incompatible changes are made to SDP, the major version number will be incremented. This guarantees that if a client and a server support a common major version number, they can communicate if each uses only features of the specification with a minor version number that is supported by both client and server.



5.2.4 ServiceDatabaseState Attribute

Attribute Name	Attribute ID	Attribute Value Type
ServiceDatabaseState	0x0201	32-bit unsigned integer

Description:

The ServiceDatabaseState is a 32-bit integer that is used to facilitate caching of service records. If this attribute exists, its value is guaranteed to change when any of the other service records are added to or deleted from the server's database. If this value has not changed since the last time a client queried its value, the client knows that a) none of the other service records maintained by the SDP server have been added or deleted; and b) any service record handles acquired from the server are still valid. A client should query this attribute's value when a connection to the server is established, prior to using any service record handles acquired during a previous connection.

Note that the ServiceDatabaseState attribute does not change when existing service records are modified, including the addition, removal, or modification of service attributes. A service record's ServiceRecordState attribute indicates when that service record is modified.

5.2.5 Reserved Attribute IDs

Attribute IDs in the range of 0x0202-0x02FF are reserved.

5.3 BROWSEGROUPDESCRIPTOR SERVICE CLASS ATTRIBUTE DEFINITIONS

This service class describes the ServiceRecord provided for each BrowseGroupDescriptor service offered on a Bluetooth device. The attributes listed in this section are only valid if the ServiceClassIDList attribute contains the BrowseGroupDescriptorServiceClassID. Note that all of the universal attributes may be included in service records of the BrowseGroupDescriptor class.

5.3.1 ServiceClassIDList Attribute

Described in the universal attribute definition for ServiceClassIDList.

Value

A UUID representing the BrowseGroupDescriptorServiceClassID.



5.3.2 GroupID Attribute

Attribute Name	Attribute ID	Attribute Value Type
GroupID	0x0200	UUID

Description:

This attribute contains a UUID that can be used to locate services that are members of the browse group that this service record describes.

5.3.3 Reserved Attribute IDs

Attribute IDs in the range of 0x0201-0x02FF are reserved.

APPENDIX A – BACKGROUND INFORMATION

A.1. Service Discovery

As computing continues to move to a network-centric model, finding and making use of services that may be available in the network becomes increasingly important. Services can include common ones such as printing, paging, FAX-ing, and so on, as well as various kinds of information access such as teleconferencing, network bridges and access points, eCommerce facilities, and so on — most any kind of service that a server or service provider might offer. In addition to the need for a standard way of discovering available services, there are other considerations: getting access to the services (finding and obtaining the protocols, access methods, “drivers” and other code necessary to utilize the service), controlling access to the services, advertising the services, choosing among competing services, billing for services, and so on. This problem is widely recognized; many companies, standards bodies and consortia are addressing it at various levels in various ways. Service Location Protocol (SLP), Jini™, and Salutation™, to name just a few, all address some aspect of service discovery.

A.2. Bluetooth Service Discovery

Bluetooth Service Discovery Protocol (SDP) addresses service discovery specifically for the Bluetooth environment. It is optimized for the highly dynamic nature of Bluetooth communications. SDP focuses primarily on discovering services available from or through Bluetooth devices. SDP does not define methods for accessing services; once services are discovered with SDP, they can be accessed in various ways, depending upon the service. This might include the use of other service discovery and access mechanisms such as those mentioned above; SDP provides a means for other protocols to be used along with SDP in those environments where this can be beneficial. While SDP can coexist with other service discovery protocols, it does not require them. In Bluetooth environments, services can be discovered using SDP and can be accessed using other protocols defined by Bluetooth.



APPENDIX B – EXAMPLE SDP TRANSACTIONS

The following are simple examples of typical SDP transactions. These are meant to be illustrative of SDP flows. The examples do not consider:

- Caching (in a caching system, the SDP client would make use of the ServiceRecordState and ServiceDatabaseState attributes);
- Service availability (if this is of interest, the SDP client should use the ServiceAvailability and/or ServiceTimeToLive attributes);
- SDP versions (the VersionNumberList attribute could be used to determine compatible SDP versions);
- SDP Error Responses (an SDP error response is possible for any SDP request that is in error); and
- Communication connection (the examples assume that an L2CAP connection is established).

The examples are meant to be illustrative of the protocol. The format used is `ObjectName[ObjectSizeInBytes] {SubObjectDefinitions}`, but this is not meant to illustrate an interface. The `ObjectSizeInBytes` is the size of the object in decimal. The `SubObjectDefinitions` (inside of `{}` characters) are components of the immediately enclosing object. Hexadecimal values shown as lower-case letters, such as for transaction IDs and service handles, are variables (the particular value is not important for the illustration, but each such symbol always represents the same value). Comments are included in this manner: `/* comment text */`.

Numeric values preceded by “0x” are hexadecimal, while those preceded by “0b” are binary. All other numeric values are decimal.

B.1. SDP Example 1 – ServiceSearchRequest

The first example is that of an SDP client searching for a generic printing service. The client does not specify a particular type of printing service. In the example, the SDP server has two available printing services. The transaction illustrates:

1. SDP client to SDP server: `SDP_ServiceSearchRequest`, specifying the `PrinterServiceClassID` (represented as a `DataElement` with a 32-bit UUID value of `ppp . . . ppp`) as the only element of the `ServiceSearchPattern`. The `PrinterServiceClassID` is assumed to be a 32-bit UUID and the data element type for it is illustrated. The `TransactionID` is illustrated as `tttt`.
2. SDP server to SDP client: `SDP_ServiceSearchResponse`, returning handles to two printing services, represented as `qqqqqqqq` for the first printing service and `rrrrrrrr` for the second printing service. The `TransactionID` is the same value as supplied by the SDP client in the corresponding request (`tttt`).

Service Discovery Protocol (SDP)

```

/* Sent from SDP Client to SDP server */
SDP_ServiceSearchRequest[15] {
    PDUID[1] {
        0x02
    }
    TransactionID[2] {
        0xtttt
    }
    ParameterLength[2] {
        0x000A
    }
    ServiceSearchPattern[7] {
        DataElementSequence[7] {
            0b00110 0b101 0x05
            UUID[5] {
                /* PrinterServiceClassID */
                0b00011 0b010 0xppppppppp
            }
        }
    }
    MaximumServiceRecordCount[2] {
        0x0003
    }
    ContinuationState[1] {
        /* no continuation state */
        0x00
    }
}

```

```

/* Sent from SDP server to SDP client */
SDP_ServiceSearchResponse[18] {
    PDUID[1] {
        0x03
    }
    TransactionID[2] {
        0xtttt
    }
    ParameterLength[2] {
        0x000D
    }
    TotalServiceRecordCount[2] {
        0x0002
    }
    CurrentServiceRecordCount[2] {
        0x0002
    }
    ServiceRecordHandleList[8] {
        /* print service 1 handle */
        0xqqqqqqqq
        /* print service 2 handle */
        0xrrrrrrrr
    }
    ContinuationState[1] {
        /* no continuation state */
    }
}

```



```

    0x00
  }
}

```

B.2. SDP Example 2 – ServiceAttributeTransaction

The second example continues the first example. In Example 1, the SDP client obtained handles to two printing services. In Example 2, the client uses one of those service handles to obtain the ProtocolDescriptorList attribute for that printing service. The transaction illustrates:

1. SDP client to SDP server: SDP_ServiceAttributeRequest, presenting the previously obtained service handle (the one denoted as `qqqqqqqq`) and specifying the ProtocolDescriptorList attribute ID (AttributeID `0x0004`) as the only attribute requested (other attributes could be retrieved in the same transaction if desired). The TransactionID is illustrated as `uuuu` to distinguish it from the TransactionID of Example 1.
2. SDP server to SDP client: SDP_ServiceAttributeResponse, returning the ProtocolDescriptorList for the specified printing service. This protocol stack is assumed to be (L2CAP), (RFCOMM, 2), (PostscriptStream)). The ProtocolDescriptorList is a data element sequence in which each element is, in turn, a data element sequence whose first element is a UUID representing the protocol, and whose subsequent elements are protocol-specific parameters. In this example, one such parameter is included for the RFCOMM protocol, an 8-bit value indicating RFCOMM server channel 2. The Transaction ID is the same value as supplied by the SDP client in the corresponding request (`uuuu`). The Attributes returned are illustrated as a data element sequence where the protocol descriptors are 32-bit UUIDs and the RFCOMM server channel is a data element with an 8-bit value of 2.

```

/* Sent from SDP Client to SDP server */
SDP_ServiceAttributeRequest[17] {
  PDUID[1] {
    0x04
  }
  TransactionID[2] {
    0xuuuu
  }
  ParameterLength[2] {
    0x000C
  }
  ServiceRecordHandle[4] {
    0xqqqqqqqq
  }
  MaximumAttributeByteCount[2] {
    0x0080
  }
  AttributeIDList[5] {
    DataElementSequence[5] {

```

Service Discovery Protocol (SDP)

```

        0b00110 0b101 0x03
        AttributeID[3] {
            0b00001 0b001 0x0004
        }
    }
}
ContinuationState[1] {
    /* no continuation state */
    0x00
}
}

/* Sent from SDP server to SDP client */
SDP_ServiceAttributeResponse[38] {
    PDUID[1] {
        0x05
    }
    TransactionID[2] {
        0xuuuu
    }
    ParameterLength[2] {
        0x0021
    }
    AttributeListByteCount[2] {
        0x001E
    }
    AttributeList[30] {
        DataElementSequence[30] {
            0b00110 0b101 0x1C
            Attribute[28] {
                AttributeID[3] {
                    0b00001 0b001 0x0004
                }
                AttributeValue[25] {
                    /* ProtocolDescriptorList */
                    DataElementSequence[25] {
                        0b00110 0b101 0x17
                        /* L2CAP protocol descriptor */
                        DataElementSequence[7] {
                            0b00110 0b101 0x05
                            UUID[5] {
                                /* L2CAP Protocol UUID */
                                0b00011 0b010 <32-bit L2CAP UUID>
                            }
                        }
                    }
                    /* RFCOMM protocol descriptor */
                    DataElementSequence[9] {
                        0b00110 0b101 0x07
                        UUID[5] {
                            /* RFCOMM Protocol UUID */
                            0b00011 0b010 <32-bit RFCOMM UUID>
                        }
                    }
                    /* parameter for server 2 */
                    Uint8[2] {
                        0b00001 0b000 0x02
                    }
                }
            }
        }
    }
}

```



```

/* PostscriptStream protocol descriptor */
DataElementSequence[7] {
    0b00110 0b101 0x05
    UUID[5] {
        /* PostscriptStream Protocol UUID */
        0b00011 0b010 <32-bit PostscriptStream UUID>
    }
}
}
}
}
}
}
ContinuationState[1] {
    /* no continuation state */
    0x00
}
}

```

B.3. SDP Example 3 – ServiceSearchAttributeTransaction

The third example is a form of service browsing, although it is not generic browsing in that it does not make use of SDP browse groups. Instead, an SDP client is searching for available Synchronization services that can be presented to the user for selection. The SDP client does not specify a particular type of synchronization service. In the example, the SDP server has three available synchronization services: an address book synchronization service and a calendar synchronization service (both from the same provider), and a second calendar synchronization service from a different provider. The SDP client is retrieving the same attributes for each of these services; namely, the data formats supported for the synchronization service (vCard, vCal, iCal, etc.) and those attributes that are relevant for presenting information to the user about the services. Also assume that the maximum size of a response is 400 bytes. Since the result is larger than this, the SDP client will repeat the request supplying a continuation state parameter to retrieve the remainder of the response. The transaction illustrates:

1. SDP client to SDP server: SDP_ServiceSearchAttributeRequest, specifying the generic SynchronisationServiceClassID (represented as a data element whose 32-bit UUID value is *sss . . . sss*) as the only element of the ServiceSearchPattern. The SynchronisationServiceClassID is assumed to be a 32-bit UUID. The requested attributes are the ServiceRecordHandle (attribute ID 0x0000), ServiceClassIDList (attribute ID 0x0001), IconURL (attribute ID 0x000C), ServiceName (attribute ID 0x0100), ServiceDescription (attribute ID 0x0101), and ProviderName (attributeID 0x0102) attributes; as well as the service-specific SupportedDataStores (AttributeID 0x0301). Since the first two attribute IDs (0x0000 and 0x0001) and three other attribute IDs (0x0100, 0x0101, and 0x0102) are consecutive, they are specified as attribute ranges. The TransactionID is illustrated as *vvvv* to distinguish it from the TransactionIDs of the other Examples.



Note that values in the service record's primary language are requested for the text attributes (ServiceName, ServiceDescription and ProviderName) so that absolute attribute IDs may be used, rather than adding offsets to a base obtained from the LanguageBaseAttributeIDList attribute.

2. SDP server to SDP client: SDP_ServiceSearchAttributeResponse, returning the specified attributes for each of the three synchronization services. In the example, each ServiceClassIDList is assumed to contain a single element, the generic SynchronisationServiceClassID (a 32-bit UUID represented as sss...sss). Each of the other attributes contain illustrative data in the example (the strings have illustrative text; the icon URLs are illustrative, for each of the respective three synchronization services; and the SupportedDataStore attribute is represented as an unsigned 8-bit integer where 0x01 = vCard2.1, 0x02 = vCard3.0, 0x03 = vCal1.0 and 0x04 = iCal). Note that one of the service records (the third for which data is returned) has no ServiceDescription attribute. The attributes are returned as a data element sequence, where each element is in turn a data element sequence representing a list of attributes. Within each attribute list, the ServiceClassIDList is a data element sequence while the remaining attributes are single data elements. The Transaction ID is the same value as supplied by the SDP client in the corresponding request (0xvvvv). Since the entire result cannot be returned in a single response, a non-null continuation state is returned in this first response.

Note that the total length of the initial data element sequence (487 in the example) is indicated in the first response, even though only a portion of this data element sequence (368 bytes in the example, as indicated in the AttributeLists byte count) is returned in the first response. The remainder of this data element sequence is returned in the second response (without an additional data element header).

3. SDP client to SDP server: SDP_ServiceSearchAttributeRequest, with the same parameters as in step 1, except that the continuation state received from the server in step 2 is included as a request parameter. The TransactionID is changed to 0xwww to distinguish it from previous request.
4. SDP server to SDP client: SDP_ServiceSearchAttributeResponse, with the remainder of the result computed in step 2 above. Since all of the remaining result fits in this second response, a null continuation state is included.

```
/* Part 1 -- Sent from SDP Client to SDP server */
SdpSDP_ServiceSearchAttributeRequest[33] {
  PDUID[1] {
    0x06
  }
  TransactionID[2] {
    0xvvvv
  }
  ParameterLength[2] {
    0x001B
  }
  ServiceSearchPattern[7] {
```

Service Discovery Protocol (SDP)

```

    DataElementSequence[7] {
        0b00110 0b101 0x05
        UUID[5] {
            /* SynchronisationServiceClassID */
            0b00011 0b010 0xssssssss
        }
    }
}
MaximumAttributeByteCount[2] {
    0x0190
}
AttributeIDList[18] {
    DataElementSequence[18] {
        0b00110 0b101 0x10
        AttributeIDRange[5] {
            0b00001 0b010 0x00000001
        }
        AttributeID[3] {
            0b00001 0b001 0x000C
        }
        AttributeIDRange[5] {
            0b00001 0b010 0x01000102
        }
        AttributeID[3] {
            0b00001 0b001 0x0301
        }
    }
}
ContinuationState[1] {
    /* no continuation state */
    0x00
}

/* Part 2 -- Sent from SDP server to SDP client */
SdpSDP_ServiceSearchAttributeResponse[384] {
    PDUID[1] {
        0x07
    }
    TransactionID[2] {
        0xvvvv
    }
    ParameterLength[2] {
        0x017B
    }
    AttributeListByteCount[2] {
        0x0170
    }
    AttributeLists[368] {
        DataElementSequence[487] {
            0b00110 0b110 0x01E4
            DataElementSequence[178] {
                0b00110 0b101 0xB0
                Attribute[8] {
                    AttributeID[3] {
                        0b00001 0b001 0x0000
                    }
                    AttributeValue[5] {

```

Service Discovery Protocol (SDP)

```

        /* service record handle */
        0b00001 0b010 0xhhhhhhhh
    }
}
Attribute[10] {
    AttributeID[3] {
        0b00001 0b001 0x0001
    }
    AttributeValue[7] {
        DataElementSequence[7] {
            0b00110 0b101 0x05
            UUID[5] {
                /* SynchronisationServiceClassID */
                0b00011 0b010 0xssssssss
            }
        }
    }
}
Attribute[35] {
    AttributeID[3] {
        0b00001 0b001 0x000C
    }
    AttributeValue[32] {
        /* IconURL; '*' replaced by client application */
        0b01000 0b101 0x1E
        "http://Synchronisation/icons/*"
    }
}
Attribute[22] {
    AttributeID[3] {
        0b00001 0b001 0x0100
    }
    AttributeValue[19] {
        /* service name */
        0b00100 0b101 0x11
        "Address Book Sync"
    }
}
Attribute[59] {
    AttributeID[3] {
        0b00001 0b001 0x0101
    }
    AttributeValue[56] {
        /* service description */
        0b00100 0b101 0x36
        "Synchronisation Service for"
        " vCard Address Book Entries"
    }
}
Attribute[37] {
    AttributeID[3] {
        0b00001 0b001 0x0102
    }
    AttributeValue[34] {
        /* service provider */
        0b00100 0b101 0x20
        "Synchronisation Specialists Inc."
    }
}

```



```

    }
  }
  Attribute[5] {
    AttributeID[3] {
      0b00001 0b001 0x0301
    }
    AttributeValue[2] {
      /* Supported Data Store 'phonebook' */
      0b00001 0b000 0x01
    }
  }
}
DataElementSequence[175] {
  0b00110 0b101 0xAD
  Attribute[8] {
    AttributeID[3] {
      0b00001 0b001 0x0000
    }
    AttributeValue[5] {
      /* service record handle */
      0b00001 0b010 0xmmmmmmmm
    }
  }
}
Attribute[10] {
  AttributeID[3] {
    0b00001 0b001 0x0001
  }
  AttributeValue[7] {
    DataElementSequence[7] {
      0b00110 0b101 0x05
      UUID[5] {
        /* SynchronisationServiceClassID */
        0b00011 0b010 0xssssssss
      }
    }
  }
}
Attribute[35] {
  AttributeID[3] {
    0b00001 0b001 0x000C
  }
  AttributeValue[32] {
    /* IconURL; '*' replaced by client application */
    0b01000 0b101 0x1E
    "http://Synchronisation/icons/*"
  }
}
Attribute[21] {
  AttributeID[3] {
    0b00001 0b001 0x0100
  }
  AttributeValue[18] {
    /* service name */
    0b00100 0b101 0x10
    "Appointment Sync"
  }
}
}

```

Service Discovery Protocol (SDP)

```

        Attribute[57] {
            AttributeID[3] {
                0b00001 0b001 0x0101
            }
            AttributeValue[54] {
                /* service description */
                0b00100 0b101 0x34
                "Synchronisation Service for"
                " vCal Appointment Entries"
            }
        }
        Attribute[37] {
            AttributeID[3] {
                0b00001 0b001 0x0102
            }
            AttributeValue[34] {
                /* service provider */
                0b00100 0b101 0x20
                "Synchronisation Specialists Inc."
            }
        }
        Attribute[5] {
            AttributeID[3] {
                0b00001 0b001 0x0301
            }
            AttributeValue[2] {
                /* Supported Data Store 'calendar' */
                0b00001 0b000 0x03
            }
        }
    }
    /* } Data element sequence of attribute lists */
    /* is not completed in this PDU. */
}
ContinuationState[9] {
    /* 8 bytes of continuation state */
    0x08 0xxxxxxxxxxxxxxxxxxx
}

/* Part 3 -- Sent from SDP Client to SDP server */
SdpSDP_ServiceSearchAttributeRequest[41] {
    PDUID[1] {
        0x06
    }
    TransactionID[2] {
        0xwww
    }
    ParameterLength[2] {
        0x0024
    }
    ServiceSearchPattern[7] {
        DataElementSequence[7] {
            0b00110 0b101 0x05
            UUID[5] {
                /* SynchronisationServiceClassID */
                0b00011 0b010 0xxxxxxxx
            }
        }
    }
}

```

Service Discovery Protocol (SDP)

```

    }
  }
}
MaximumAttributeByteCount[2] {
  0x0180
}
AttributeIDList[18] {
  DataElementSequence[18] {
    0b00110 0b101 0x10
    AttributeIDRange[5] {
      0b00001 0b010 0x00000001
    }
    AttributeID[3] {
      0b00001 0b001 0x000C
    }
    AttributeIDRange[5] {
      0b00001 0b010 0x01000102
    }
    AttributeID[3] {
      0b00001 0b001 0x0301
    }
  }
}
ContinuationState[9] {
  /* same 8 bytes of continuation state */
  /* received in part 2 */
  0x08 0xzzzzzzzzzzzzzzzz
}
}

```

Part 4 -- Sent from SDP server to SDP client

```

SdpSDP_ServiceSearchAttributeResponse[115] {
  PDUID[1] {
    0x07
  }
  TransactionID[2] {
    0xwww
  }
  ParameterLength[2] {
    0x006E
  }
  AttributeListByteCount[2] {
    0x006B
  }
  AttributeLists[107] {
    /* Continuing the data element sequence of */
    /* attribute lists begun in Part 2. */
    DataElementSequence[107] {
      0b00110 0b101 0x69
      Attribute[8] {
        AttributeID[3] {
          0b00001 0b001 0x0000
        }
        AttributeValue[5] {
          /* service record handle */
          0b00001 0b010 0xffffffff
        }
      }
    }
  }
}

```



```

    }
  }
  Attribute[10] {
    AttributeID[3] {
      0b00001 0b001 0x0001
    }
    AttributeValue[7] {
      DataElementSequence[7] {
        0b00110 0b101 0x05
        UUID[5] {
          /* SynchronisationServiceClassID */
          0b00011 0b010 0xssssssss
        }
      }
    }
  }
}
Attribute[35] {
  AttributeID[3] {
    0b00001 0b001 0x000C
  }
  AttributeValue[32] {
    /* IconURL; '*' replaced by client application */
    0b01000 0b101 0x1E
    "http://DevManufacturer/icons/*"
  }
}
Attribute[18] {
  AttributeID[3] {
    0b00001 0b001 0x0100
  }
  AttributeValue[15] {
    /* service name */
    0b00100 0b101 0x0D
    "Calendar Sync"
  }
}
Attribute[29] {
  AttributeID[3] {
    0b00001 0b001 0x0102
  }
  AttributeValue[26] {
    /* service provider */
    0b00100 0b101 0x18
    "Device Manufacturer Inc."
  }
}
Attribute[5] {
  AttributeID[3] {
    0b00001 0b001 0x0301
  }
  AttributeValue[2] {
    /* Supported Data Store 'calendar' */
    0b00001 0b000 0x03
  }
}
}
/* This completes the data element sequence */

```

Service Discovery Protocol (SDP)

```
    /* of attribute lists begun in Part 2.
    */
    ContinuationState[1] {
        /* no continuation state */
        0x00
    }
}
```

GENERIC ACCESS PROFILE



This profile defines the generic procedures related to discovery of Bluetooth devices (idle mode procedures) and link management aspects of connecting to Bluetooth devices (connecting mode procedures). It also defines procedures related to use of different security levels. In addition, this profile includes common format requirements for parameters accessible on the user interface level.





CONTENTS

1	Introduction	179
1.1	Scope	179
1.2	Symbols and conventions	179
1.2.1	Requirement status symbols	179
1.2.2	Signaling diagram conventions	180
1.2.3	Notation for timers and counters	180
2	Profile overview	181
2.1	Profile stack	181
2.2	Configurations and roles	181
2.3	User requirements and scenarios	182
2.4	Profile fundamentals	183
2.5	Conformance	183
3	User interface aspects	185
3.1	The user interface level	185
3.2	Representation of Bluetooth parameters	185
3.2.1	Bluetooth device address (BD_ADDR)	185
3.2.1.1	Definition	185
3.2.1.2	Term on user interface level	185
3.2.1.3	Representation	185
3.2.2	Bluetooth device name (the user-friendly name)	185
3.2.2.1	Definition	185
3.2.2.2	Term on user interface level	186
3.2.2.3	Representation	186
3.2.3	Bluetooth passkey (Bluetooth PIN)	186
3.2.3.1	Definition	186
3.2.3.2	Terms at user interface level	186
3.2.3.3	Representation	186
3.2.4	Class of Device	187
3.2.4.1	Definition	187
3.2.4.2	Term on user interface level	188
3.2.4.3	Representation	188
3.3	Pairing	188
4	Modes	189
4.1	Discoverability modes	189
4.1.1	Non-discoverable mode	190
4.1.1.1	Definition	190
4.1.1.2	Term on UI-level	190
4.1.2	Limited discoverable mode	190
4.1.2.1	Definition	190



4.1.2.2	Conditions.....	191
4.1.2.3	Term on UI-level	191
4.1.3	General discoverable mode	191
4.1.3.1	Definition.....	191
4.1.3.2	Conditions.....	192
4.1.3.3	Term on UI-level	192
4.2	Connectability modes.....	193
4.2.1	Non-connectable mode	193
4.2.1.1	Definition.....	193
4.2.1.2	Term on UI-level	193
4.2.2	Connectable mode	193
4.2.2.1	Definition.....	193
4.2.2.2	Term on UI-level	194
4.3	Pairing modes.....	195
4.3.1	Non-pairable mode.....	195
4.3.1.1	Definition.....	195
4.3.1.2	Term on UI-level	195
4.3.2	Pairable mode.....	195
4.3.2.1	Definition.....	195
4.3.2.2	Term on UI-level	195
5	Security aspects.....	197
5.1	Authentication	197
5.1.1	Purpose.....	197
5.1.2	Term on UI level	197
5.1.3	Procedure	198
5.1.4	Conditions	198
5.2	Security modes	198
5.2.1	Security mode 1 (non-secure).....	200
5.2.2	Security mode 2 (service level enforced security).....	200
5.2.3	Security modes 3 (link level enforced security).....	200
6	Idle mode procedures.....	201
6.1	General inquiry	201
6.1.1	Purpose.....	201
6.1.2	Term on UI level	201
6.1.3	Description	202
6.1.4	Conditions	202
6.2	Limited inquiry.....	202
6.2.1	Purpose.....	202
6.2.2	Term on UI level	203
6.2.3	Description	203



6.2.4	Conditions	203
6.3	Name discovery	204
6.3.1	Purpose	204
6.3.2	Term on UI level	204
6.3.3	Description	204
6.3.3.1	Name request	204
6.3.3.2	Name discovery	204
6.3.4	Conditions	205
6.4	Device discovery	205
6.4.1	Purpose	205
6.4.2	Term on UI level	205
6.4.3	Description	206
6.4.4	Conditions	206
6.5	Bonding	206
6.5.1	Purpose	206
6.5.2	Term on UI level	206
6.5.3	Description	207
6.5.3.1	General bonding	207
6.5.3.2	Dedicated bonding	208
6.5.4	Conditions	208
7	Establishment procedures	209
7.1	Link establishment	209
7.1.1	Purpose	209
7.1.2	Term on UI level	209
7.1.3	Description	210
7.1.3.1	B in security mode 1 or 2	210
7.1.3.2	B in security mode 3	211
7.1.4	Conditions	211
7.2	Channel establishment	212
7.2.1	Purpose	212
7.2.2	Term on UI level	212
7.2.3	Description	212
7.2.3.1	B in security mode 2	213
7.2.3.2	B in security mode 1 or 3	213
7.2.4	Conditions	213
7.3	Connection establishment	214
7.3.1	Purpose	214
7.3.2	Term on UI level	214
7.3.3	Description	214
7.3.3.1	B in security mode 2	214



7.3.3.2	B in security mode 1 or 3	215
7.3.4	Conditions	215
7.4	Establishment of additional connection.....	215
8	Definitions	217
8.1	General definitions	217
8.2	Connection-related definitions	217
8.3	Device-related definitions	218
8.4	Procedure-related definitions	219
8.5	Security-related definitions	219
9	Appendix A (Normative): Timers and constants	221
10	Appendix B (Informative): Information flows of related procedures	223
10.1	Imp-authentication	223
10.2	Imp-pairing	224
10.3	Service discovery	225
11	References	227



FOREWORD

Interoperability between devices from different manufacturers is provided for a specific service and use case, if the devices conform to a Bluetooth SIG-defined profile specification. A profile defines a selection of messages and procedures (generally termed *capabilities*) from the Bluetooth SIG specifications and gives a description of the air interface for specified service(s) and use case(s).

All defined features are process-mandatory. This means that, if a feature is used, it is used in a specified manner. Whether the provision of a feature is mandatory or optional is stated separately for both sides of the Bluetooth air interface.



1 INTRODUCTION

1.1 SCOPE

The purpose of the Generic Access Profile is:

To introduce definitions, recommendations and common requirements related to modes and access procedures that are to be used by transport and application profiles.

To describe how devices are to behave in standby and connecting states in order to guarantee that links and channels always can be established between Bluetooth devices, and that multi-profile operation is possible. Special focus is put on discovery, link establishment and security procedures.

To state requirements on user interface aspects, mainly coding schemes and names of procedures and parameters, that are needed to guarantee a satisfactory user experience.

1.2 SYMBOLS AND CONVENTIONS

1.2.1 Requirement status symbols

In this document (especially in the profile requirements tables), the following symbols are used:

'M' for mandatory to support (used for capabilities that shall be used in the profile);

'O' for optional to support (used for capabilities that can be used in the profile);

'C' for conditional support (used for capabilities that shall be used in case a certain other capability is supported);

'X' for excluded (used for capabilities that may be supported by the unit but shall never be used in the profile);

'N/A' for not applicable (in the given context it is impossible to use this capability).

Some excluded capabilities are capabilities that, according to the relevant Bluetooth specification, are mandatory. These are features that may degrade operation of devices following this profile. Therefore, these features shall never be activated while a unit is operating as a unit within this profile.

In this specification, the word *shall* is used for mandatory requirements, the word *should* is used to express recommendations and the word *may* is used for options.



1.2.2 Signaling diagram conventions

The following arrows are used in diagrams describing procedures

:

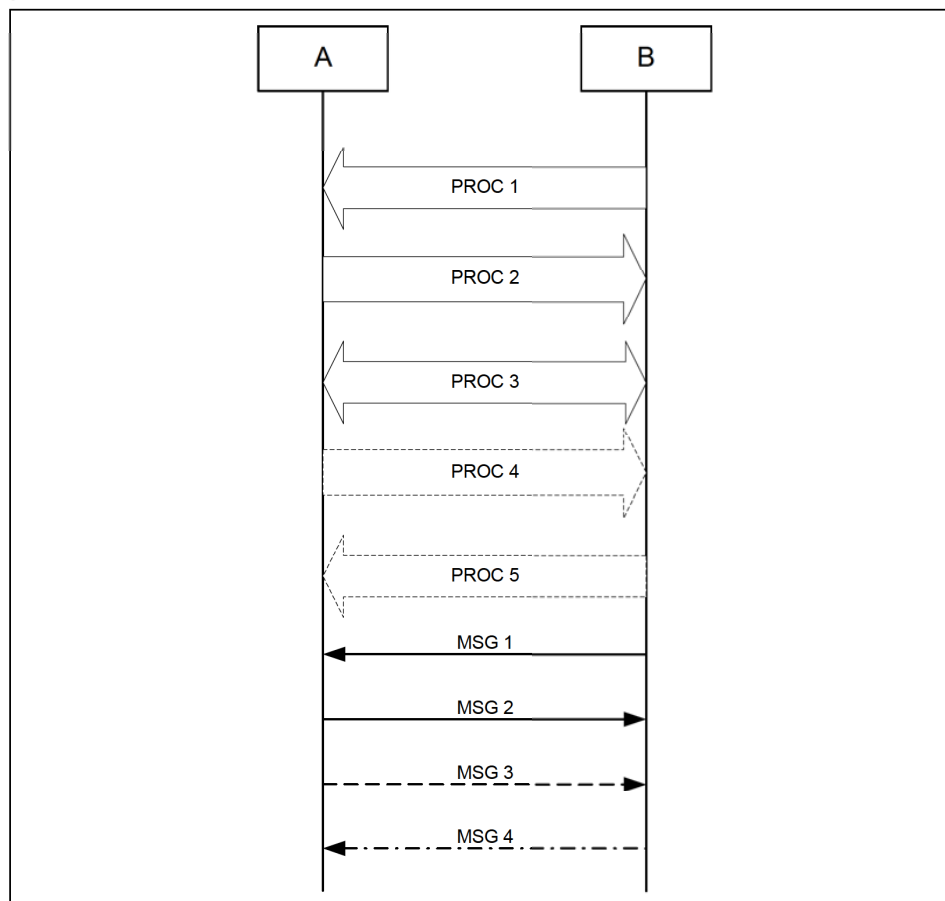


Figure 1.1: Arrows used in signaling diagrams

In the figure above, the following cases are shown: PROC1 is a sub-procedure initiated by B. PROC2 is a sub-procedure initiated by A. PROC3 is a sub-procedure where the initiating side is undefined (may be both A or B). Dashed arrows denote optional steps. PROC4 indicates an optional sub-procedure initiated by A, and PROC5 indicates an optional sub-procedure initiated by B.

MSG1 is a message sent from B to A. MSG2 is a message sent from A to B. MSG3 indicates an optional message from A to B, and MSG4 indicates a conditional message from B to A.

1.2.3 Notation for timers and counters

Timers are introduced specific to this profile. To distinguish them from timers used in the Bluetooth protocol specifications and other profiles, these timers are named in the following format: 'T_{GAP}(nnn)'.

2 PROFILE OVERVIEW

2.1 PROFILE STACK

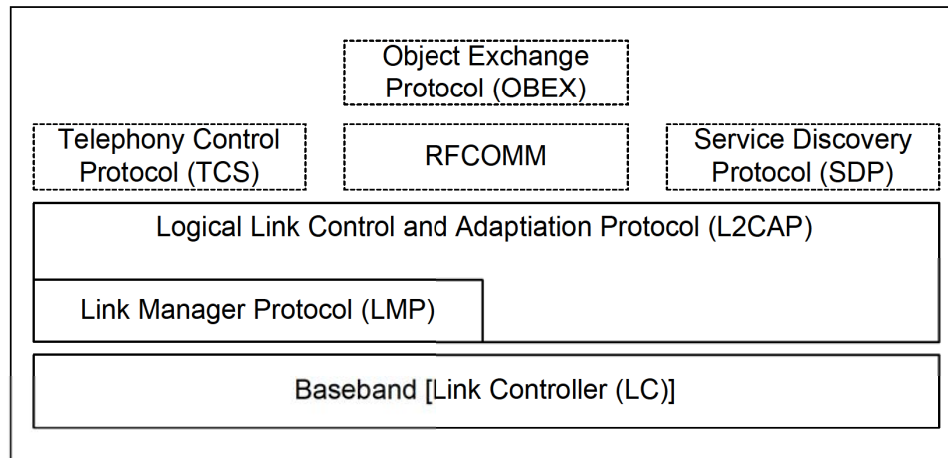


Figure 2.1: Profile stack covered by this profile.

The main purpose of this profile is to describe the use of the lower layers of the Bluetooth protocol stack (LC and LMP). To describe security related alternatives, also higher layers (L2CAP, RFCOMM and OBEX) are included.

2.2 CONFIGURATIONS AND ROLES

For the descriptions in this profile of the roles that the two devices involved in a Bluetooth communication can take, the generic notation of the A-party (the *paging device* in case of link establishment, or *initiator* in case of another procedure on an established link) and the B-party (*paged device* or *acceptor*) is used. The A-party is the one that, for a given procedure, initiates the establishment of the physical link or initiates a transaction on an existing link.

This profile handles the procedures between two devices related to discovery and connecting (link and connection establishment) for the case where none of the two devices has any link established as well as the case where (at least) one device has a link established (possibly to a third device) before starting the described procedure.

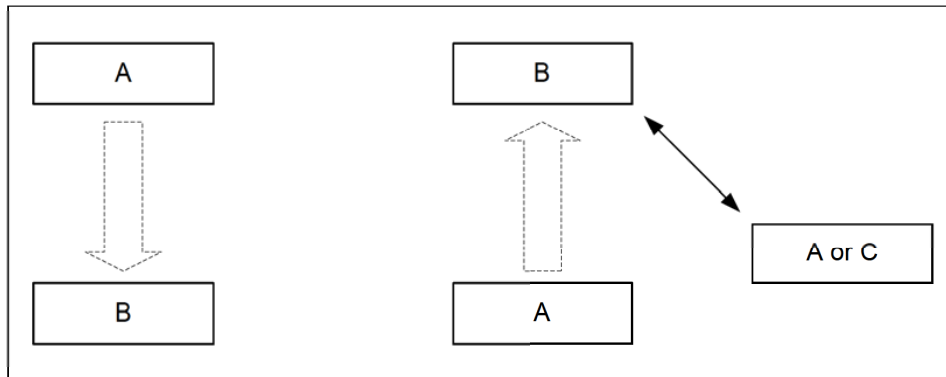


Figure 2.2: This profile covers procedures initiated by one device (A) towards another device (B) that may or may not have an existing Bluetooth link active.

The initiator and the acceptor generally operate the generic procedures according to this profile or another profile referring to this profile. If the acceptor operates according to several profiles simultaneously, this profile describes generic mechanisms for how to handle this.

2.3 USER REQUIREMENTS AND SCENARIOS

The Bluetooth user should in principle be able to connect a Bluetooth device to any other Bluetooth device. Even if the two connected devices don't share any common application, it should be possible for the user to find this out using basic Bluetooth capabilities. When the two devices do share the same application but are from different manufacturers, the ability to connect them should not be blocked just because manufacturers choose to call basic Bluetooth capabilities by different names on the user interface level or implement basic procedures to be executed in different orders.



2.4 PROFILE FUNDAMENTALS

This profile states the requirements on names, values and coding schemes used for names of parameters and procedures experienced on the user interface level.

This profile defines modes of operation that are not service- or profile-specific, but that are generic and can be used by profiles referring to this profile, and by devices implementing multiple profiles.

This profile defines the general procedures that can be used for discovering identities, names and basic capabilities of other Bluetooth devices that are in a mode where they can be discoverable. Only procedures where no channel or connection establishment is used are specified.

This profile defines the general procedure for how to create bonds (i.e. dedicated exchange of link keys) between Bluetooth devices.

This profile describes the general procedures that can be used for establishing connections to other Bluetooth devices that are in mode that allows them to accept connections and service requests.

2.5 CONFORMANCE

Bluetooth devices that do not conform to any other Bluetooth profile shall conform to this profile to ensure basic interoperability.

Bluetooth devices that conform to another Bluetooth profile may use adaptations of the generic procedures as specified by that other profile. They shall, however, be compatible with devices compliant to this profile at least on the level of the supported generic procedures.

If conformance to this profile is claimed, all capabilities indicated mandatory for this profile shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated. All mandatory capabilities, and optional and conditional capabilities for which support is indicated, are subject to verification as part of the Bluetooth certification program.



3 USER INTERFACE ASPECTS

3.1 THE USER INTERFACE LEVEL

In the context of this specification, the user interface level refers to places (such as displays, dialog boxes, manuals, packaging, advertising, etc.) where users of Bluetooth devices encounter names, values and numerical representation of Bluetooth terminology and parameters.

This profile specifies the generic terms that should be used on the user interface level.

3.2 REPRESENTATION OF BLUETOOTH PARAMETERS

3.2.1 Bluetooth device address (BD_ADDR)

3.2.1.1 Definition

BD_ADDR is the address used by a Bluetooth device as defined in [1]. It is received from a remote device during the device discovery procedure.

3.2.1.2 Term on user interface level

When the Bluetooth address is referred to on UI level, the term 'Bluetooth Device Address' should be used.

3.2.1.3 Representation

On BB level the BD_ADDR is represented as 48 bits [1].

On the UI level the Bluetooth address shall be represented as 12 hexadecimal characters, possibly divided into sub-parts separated by ':'.
(E.g., '000C3E3A4B69' or '00:0C:3E:3A:4B:69'.) At UI level, any number shall

have the MSB -> LSB (from left to right) 'natural' ordering (e.g., the number '16' shall be shown as '0x10').

3.2.2 Bluetooth device name (the user-friendly name)

3.2.2.1 Definition

The Bluetooth device name is the user-friendly name that a Bluetooth device presents itself with. It is a character string returned in LMP_name_res as response to a LMP_name_req.



3.2.2.2 Term on user interface level

When the Bluetooth device name is referred to on UI level, the term 'Bluetooth Device Name' should be used.

3.2.2.3 Representation

The Bluetooth device name can be up to 248 bytes maximum according to [2]. It shall be encoded according to UTF-8 (i.e. name entered on UI level may be down to 82 characters outside the Unicode range 0x00-0x7F are used).

A device can not expect that a general remote device is able to handle more than the first 40 characters of the Bluetooth device name. If a remote device has limited display capabilities, it may use only the first 20 characters.

3.2.3 Bluetooth passkey (Bluetooth PIN)

3.2.3.1 Definition

The Bluetooth PIN is used to authenticate two Bluetooth devices (that have not previously exchanged link keys) to each other and create a trusted relationship between them. The PIN is used in the pairing procedure (see [Section 10.2 on page 224](#)) to generate the initial link key that is used for further authentication.

The PIN may be entered on UI level but may also be stored in the device; e.g. in the case of a device without sufficient MMI for entering and displaying digits.

3.2.3.2 Terms at user interface level

When the Bluetooth PIN is referred to on UI level, the term 'Bluetooth Passkey' should be used.

3.2.3.3 Representation

The Bluetooth PIN has different representations on different levels. PINBB is used on baseband level, and PINUI is used on user interface level. PINBB is the PIN used by [1] for calculating the initialization key during the pairing procedure. PINUI is the character representation of the PIN that is entered on UI level. The transformation from PINUI to PINBB shall be according to UTF-8. According to [1], PINBB can be 128 bits (16 bytes).

PIN codes may be up to 16 characters. In order to take advantage of the full level of security all PINs should be 16 characters long. Variable PINs should be composed of alphanumeric characters chosen from within the Unicode range 0x00-0x7F. If the PIN contains any decimal digits these shall be encoded using the Unicode Basic Latin characters (i.e. code points 0x30 to 0x39) (Note 1).



For compatibility with devices with numeric keypads fixed PINs shall be composed of only decimal digits, and variable PINs may be composed of only decimal digits.

If a device supports entry of characters outside the Unicode range 0x00-0x7F other Unicode code points may be used (Note 2), except the Halfwidth and Fullwidth Forms from within the Unicode range FF00 - FFEF shall not be used (Note 3).

Examples:

User-entered code	Corresponding PIN _{BB} [0..length-1] (value as a sequence of octets in hexadecimal notation)
'0196554200906493'	length = 16, value = 0x30 0x31 0x39 0x36 0x35 0x35 0x34 0x32 0x30 0x30 0x39 0x30 0x36 0x34 0x39 0x33
'Børnelitteratur'	length = 16, value = 0x42 0xC3 0xB8 0x72 0x6e 0x65 0x6c 0x69 0x74 0x74 0x65 0x72 0x61 0x74 0x75 0x72

Note 1: This is to prevent interoperability problems since there are decimal digits at other code points (e.g. the Fullwidth digits at code points 0xff10 to 0xff19).

Note 2: Unicode characters outside the Basic Latin range (0x00 - 0x7F) encode to multiple bytes, therefore when characters outside the Basic Latin range are used the maximum number of characters in the PINUI will be less than 16. The second example illustrates a case where a 16 character string encodes to 15 bytes because the character ø is outside the Basic Latin range and encodes to two bytes (0xC3 0xB8).

Note 3: This is to prevent interoperability problems since the Halfwidth and Fullwidth forms contain alternative variants of ASCII, Katakana, Hangul, punctuation and symbols. All of the characters in the Halfwidth and Fullwidth forms have other more commonly used Unicode code points.

3.2.4 Class of Device

3.2.4.1 Definition

Class of device is a parameter received during the device discovery procedure, indicating the type of device and which types of service that are supported.



3.2.4.2 Term on user interface level

The information within the Class of Device parameter should be referred to as 'Bluetooth Device Class' (i.e. the major and minor device class fields) and 'Bluetooth Service Type' (i.e. the service class field). The terms for the defined Bluetooth Device Types and Bluetooth Service Types are defined in [11].

When using a mix of information found in the Bluetooth Device Class and the Bluetooth Service Type, the term 'Bluetooth Device Type' should be used.

3.2.4.3 Representation

The Class of device is a bit field and is defined in [11]. The UI-level representation of the information in the Class of device is implementation specific.

3.3 PAIRING

Two procedures are defined that make use of the pairing procedure defined on LMP level (LMP-pairing, see [Section 10.2 on page 224](#)). Either the user initiates the bonding procedure and enters the passkey with the explicit purpose of creating a bond (and maybe also a secure relationship) between two Bluetooth devices, or the user is requested to enter the passkey during the establishment procedure since the devices did not share a common link key beforehand. In the first case, the user is said to perform 'bonding (with entering of passkey)' and in the second case the user is said to 'authenticate using the passkey'.

4 MODES

Procedure	Ref.	Support
Discoverability modes:	4.1	
Non-discoverable mode		C1
Limited discoverable mode		O
General discoverable mode		O
Connectability modes:	4.1.3.3	
Non-connectable mode		O
Connectable mode		M
Pairing modes:	4.2.2.2	
Non-pairable mode		O
Pairable mode		C2
C1: If limited discoverable mode is supported, non-discoverable mode is mandatory, otherwise optional.		
C2: If the bonding procedure is supported, support for pairable mode is mandatory, otherwise optional.		

Table 4.1: Conformance requirements related to modes defined in this section

4.1 DISCOVERABILITY MODES

With respect to inquiry, a Bluetooth device shall be either in non-discoverable mode or in a discoverable mode. (The device shall be in one, and only one, discoverability mode at a time.) The two discoverable modes defined here are called limited discoverable mode and general discoverable mode. Inquiry is defined in [1].

When a Bluetooth device is in non-discoverable mode it does not respond to inquiry.

A Bluetooth device is said to be made discoverable, or set into a discoverable mode, when it is in limited discoverable mode or in general discoverable mode. Even when a Bluetooth device is made discoverable it may be unable to respond to inquiry due to other baseband activity [1]. A Bluetooth device that does not respond to inquiry for any of these two reasons is called a silent device.

After being made discoverable, the Bluetooth device shall be discoverable for at least $T_{GAP}(103)$.



The speed of discovery is dependent on the configuration of the inquiry scan interval and inquiry scan type of the Bluetooth device. The Host is able to configure these parameters based on trade-offs between power consumption, bandwidth and the desired speed of discovery.

4.1.1 Non-discoverable mode

4.1.1.1 Definition

When a Bluetooth device is in non-discoverable mode, it shall never enter the INQUIRY_RESPONSE state.

4.1.1.2 Term on UI-level

Bluetooth device is 'non-discoverable' or in 'non-discoverable mode'.

4.1.2 Limited discoverable mode

4.1.2.1 Definition

The limited discoverable mode should be used by devices that need to be discoverable only for a limited period of time, during temporary conditions or for a specific event. The purpose is to respond to a device that makes a limited inquiry (inquiry using the LIAC).

A Bluetooth device should not be in limited discoverable mode for more than $T_{GAP}(104)$. The scanning for the limited inquiry access code can be done either in parallel or in sequence with the scanning of the general inquiry access code. When in limited discoverable mode, one of the following options shall be used.

- *Parallel scanning*

When a Bluetooth device is in limited discoverable mode and when discovery speed is more important than power consumption or bandwidth, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(105)$ and that Interlaced Inquiry scan is used.

If, however, power consumption or bandwidth is important, but not critical, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Interlaced Inquiry scan is used.

When power consumption or bandwidth is critical it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and non-Interlaced Inquiry scan is used.

In all cases the Bluetooth device shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC and the LIAC for at least $T_{GAP}(101)$



When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the discoverability time.

- *Sequential scanning*

When a Bluetooth device is in limited discoverable mode, it shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC for at least $T_{GAP}(101)$ and enter the INQUIRY_SCAN state more often than once in $T_{GAP}(102)$ and scan for the LIAC for at least $T_{GAP}(101)$.

If an inquiry message is received when in limited discoverable mode, the entry into the INQUIRY_RESPONSE state takes precedence over the next entries into INQUIRY_SCAN state until the inquiry response is completed.

4.1.2.2 Conditions

When a device is in limited discoverable mode it shall set bit no 13 in the Major Service Class part of the Class of Device/Service field [11].

4.1.2.3 Term on UI-level

Bluetooth device is 'discoverable' or in 'discoverable mode'.

4.1.3 General discoverable mode

4.1.3.1 Definition

The general discoverable mode shall be used by devices that need to be discoverable continuously or for no specific condition. The purpose is to respond to a device that makes a general inquiry (inquiry using the GIAC).



4.1.3.2 Conditions

When a Bluetooth device is in general discoverable mode and when discovery speed is more important than power consumption or bandwidth, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(105)$ and that Interlaced Inquiry scan is used.

If, however, power consumption or bandwidth is important, but not critical, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Interlaced Inquiry scan is used.

When power consumption or bandwidth is critical it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Non-interlaced Inquiry scan is used.

In all cases the Bluetooth device shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC for at least $T_{GAP}(101)$.

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the discoverability time.

A device in general discoverable mode shall not respond to a LIAC inquiry.

4.1.3.3 Term on UI-level

Bluetooth device is 'discoverable' or in 'discoverable mode'.



4.2 CONNECTABILITY MODES

With respect to paging, a Bluetooth device shall be either in non-connectable mode or connectable mode. Paging is defined in [1].

When a Bluetooth device is in non-connectable mode it does not respond to paging. When a Bluetooth device is in connectable mode it responds to paging.

The speed of connections is dependent on the configuration of the page scan interval and page scan type of the Bluetooth device. The Host is able to configure these parameters based on trade-offs between power consumption, bandwidth and the desired speed of connection.

4.2.1 Non-connectable mode

4.2.1.1 Definition

When a Bluetooth device is in non-connectable mode it shall never enter the PAGE_SCAN state.

4.2.1.2 Term on UI-level

Bluetooth device is 'non-connectable' or in 'non-connectable mode'.

4.2.2 Connectable mode

4.2.2.1 Definition

When a Bluetooth device is in connectable mode it shall periodically enter the PAGE_SCAN state. The device makes page scan using the Bluetooth device address, BD_ADDR. Connection speed is a trade-off between power consumption / available bandwidth and speed. The Bluetooth Host is able to make these trade-offs using the Page Scan interval, Page Scan window, and Interlaced Scan parameters.

R0 page scanning should be used when connection speeds are critically important and when the paging device has a very good estimate of the Bluetooth clock. Under these conditions it is possible for paging to complete within two times the page scan window. Because the page scan interval is equal to the page scan window it is not possible for any other traffic to go over the Bluetooth link when using R0 page scanning. In R0 page scanning it is not possible to use interlaced scan. R0 page scanning is the highest power consumption mode of operation.

When connection times are critical but the other device either does not have an estimate of the Bluetooth clock or when the estimate is possibly out of date, it is better to use R1 page scanning with a very short page scan interval,



$T_{\text{GAP}}(106)$, and Interlaced scan. This configuration is also useful to achieve nearly the same connection speeds as R0 page scanning but using less power consumption and leaving bandwidth available for other connections. Under these circumstances it is possible for paging to complete within $T_{\text{GAP}}(106)$. In this case the Bluetooth device shall page scan for at least $T_{\text{GAP}}(101)$.

When connection times are important but not critical enough to sacrifice significant bandwidth and/or power consumption it is recommended to use either $T_{\text{GAP}}(107)$ or $T_{\text{GAP}}(108)$ for the scanning interval. Using Interlaced scan will reduce the connection time by half but may use twice the power consumption. Under these circumstances it is possible for paging to complete within one or two times the page scanning interval depending on whether Interlaced Scan is used. In this case the Bluetooth device shall page scan for at least $T_{\text{GAP}}(101)$.

In all cases the Bluetooth device shall enter the PAGE_SCAN state at least once in $T_{\text{GAP}}(102)$ and scan for at least $T_{\text{GAP}}(101)$.

The page scan interval, page scan window size and scan type for the six scenarios are described in the following table:

Scenario	Page Scan Interval	Page Scan Window	Scan Type
R0 (1.28s)	$T_{\text{GAP}}(107)$	$T_{\text{GAP}}(107)$	Normal scan
Fast R1 (100ms)	$T_{\text{GAP}}(106)$	$T_{\text{GAP}}(101)$	Interlaced scan
Medium R1 (1.28s)	$T_{\text{GAP}}(107)$	$T_{\text{GAP}}(101)$	Interlaced scan
Slow R1 (1.28s)	$T_{\text{GAP}}(107)$	$T_{\text{GAP}}(101)$	Normal scan
Fast R2 (2.56s)	$T_{\text{GAP}}(108)$	$T_{\text{GAP}}(101)$	Interlaced scan
Slow R2 (2.56s)	$T_{\text{GAP}}(108)$	$T_{\text{GAP}}(101)$	Normal scan

Table 4.2: Page scan parameters for connection speed scenarios

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the connection time.

4.2.2.2 Term on UI-level

Bluetooth device is 'connectable' or in 'connectable mode'.



4.3 PAIRING MODES

With respect to pairing, a Bluetooth device shall be either in non-pairable mode or in pairable mode. In pairable mode the Bluetooth device accepts pairing – i.e. creation of bonds – initiated by the remote device, and in non-pairable mode it does not. Pairing is defined in [1] and [2].

4.3.1 Non-pairable mode

4.3.1.1 Definition

When a Bluetooth device is in non-pairable mode it shall respond to a received LMP_in_rand with LMP_not_accepted with the reason *pairing not allowed*.

4.3.1.2 Term on UI-level

Bluetooth device is 'non-bondable' or in 'non-bondable mode' or "does not accept bonding".

4.3.2 Pairable mode

4.3.2.1 Definition

When a Bluetooth device is in pairable mode it shall respond to a received LMP_in_rand with LMP_accepted (or with LMP_in_rand if it has a fixed PIN).

4.3.2.2 Term on UI-level

Bluetooth device is 'bondable' or in 'bondable mode' or "accepts bonding".





5 SECURITY ASPECTS

	Procedure	Ref.	Support
1	Authentication	5.1	C1
2	Security modes	5.2	
	Security mode 1		O
	Security mode 2		C2
	Security mode 3		C2
C1: If security mode 1 is the only security mode that is supported, support for authentication is optional, otherwise mandatory. (Note: support for LMP-authentication and LMP-pairing is mandatory according [2] independent of which security mode that is used.)			
C2: If secure communication is supported, then support for at least one of Security mode 2 or Security mode 3 is mandatory.			

Table 5.1: Conformance requirements related to the generic authentication procedure and the security modes defined in this section

5.1 AUTHENTICATION

5.1.1 Purpose

The generic authentication procedure describes how the LMP-authentication and LMP-pairing procedures are used when authentication is initiated by one Bluetooth device towards another, depending on if a link key exists or not and if pairing is allowed or not.

5.1.2 Term on UI level

'Bluetooth authentication'.

5.1.3 Procedure

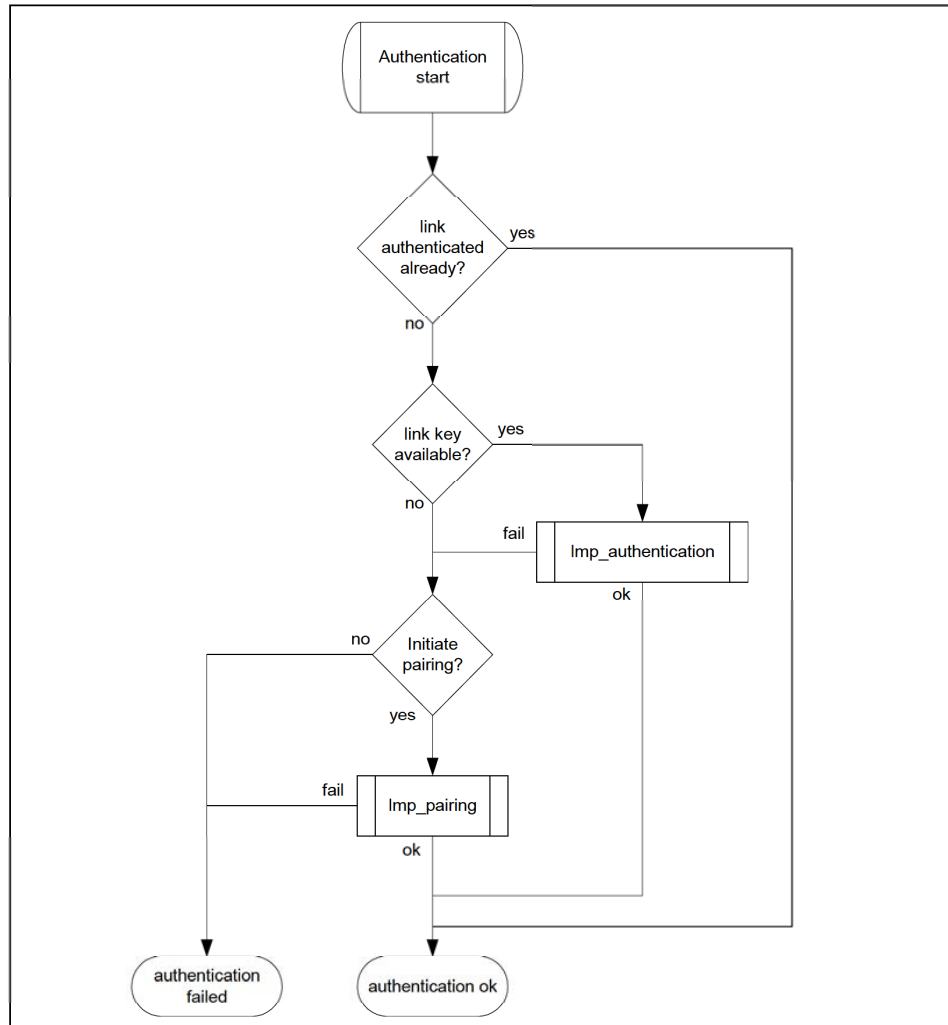


Figure 5.1: Definition of the generic authentication procedure.

5.1.4 Conditions

The device that initiates authentication has to be in security mode 2 or in security mode 3.

5.2 SECURITY MODES

The following flow chart describes where in the channel establishment procedures initiation of authentication takes place, depending on which security mode the Bluetooth device is in.

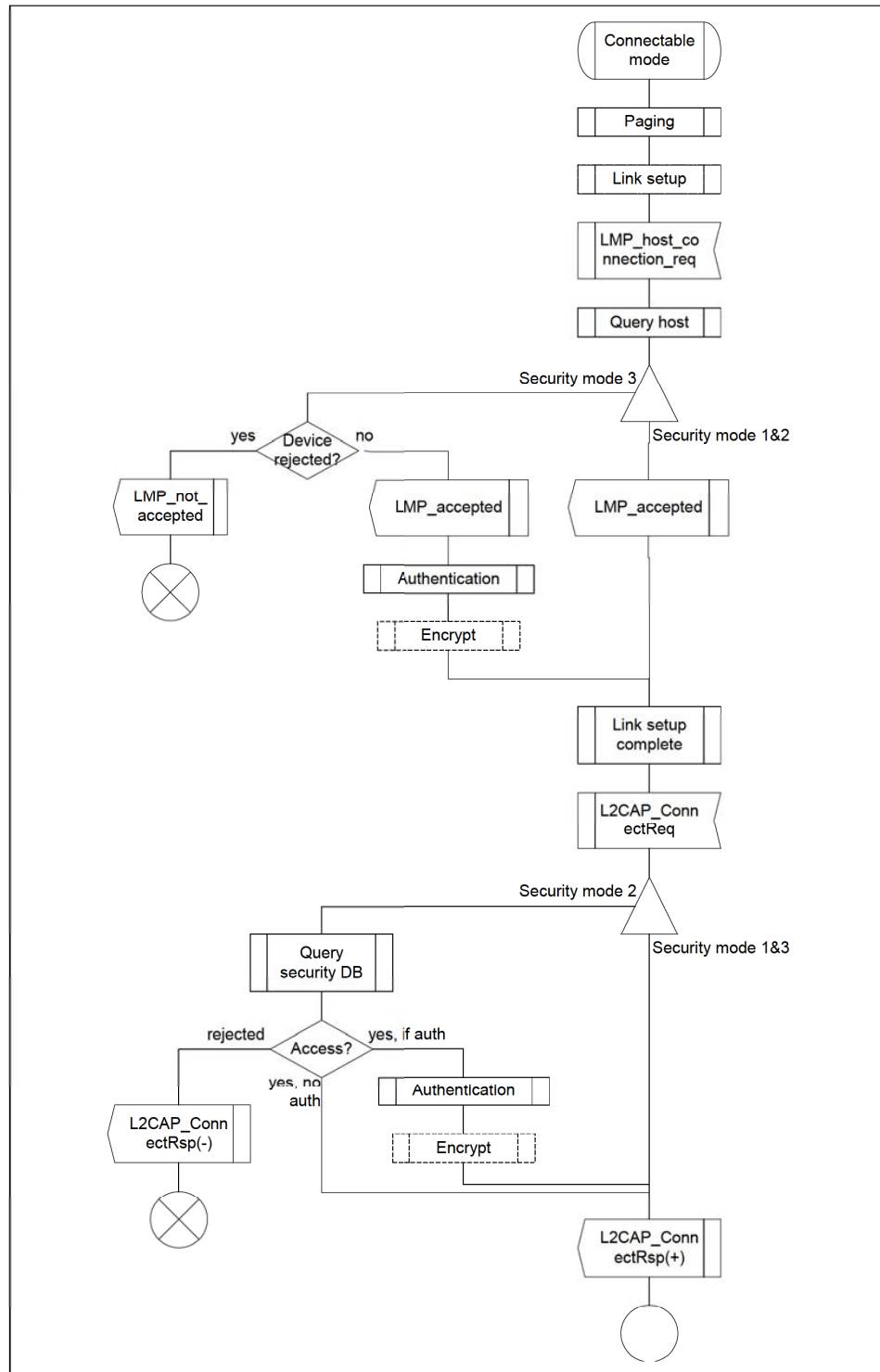


Figure 5.2: Illustration of channel establishment using different security modes.



When authentication is initiated towards a Bluetooth device, it shall act according to [2] and the current pairing mode, independent of which security mode it is in.

5.2.1 Security mode 1 (non-secure)

When a Bluetooth device is in security mode 1 it shall never initiate any security procedure (i.e., it shall never send LMP_au_rand, LMP_in_rand or LMP_encryption_mode_req).

5.2.2 Security mode 2 (service level enforced security)

When a Bluetooth device is in security mode 2 it shall not initiate any security procedure before a channel establishment request (L2CAP_ConnectReq) has been received or a channel establishment procedure has been initiated by itself. (The behavior of a device in security mode 2 is further described in [10].) Whether a security procedure is initiated or not depends on the security requirements of the requested channel or service.

A Bluetooth device in security mode 2 should classify the security requirements of its services using at least the following attributes:

- Authorization required;
- Authentication required;
- Encryption required.

Note: Security mode 1 can be considered (at least from a remote device point of view) as a special case of security mode 2 where no service has registered any security requirements.

5.2.3 Security modes 3 (link level enforced security)

When a Bluetooth device is in security mode 3 it shall initiate security procedures before it sends LMP_link_setup_complete. (The behavior of a device in security mode 3 is as described in [2].)

A Bluetooth device in security mode 3 may reject the host connection request (respond with LMP_not_accepted to the LMP_host_connection_req) based on settings in the host (e.g. only communication with pre-paired devices allowed).



6 IDLE MODE PROCEDURES

The inquiry and discovery procedures described here are applicable only to the device that initiates them (A). The requirements on the behavior of B is according to the modes specified in [Section 4 on page 189](#) and to [\[2\]](#).

	Procedure	Ref.	Support
1	General inquiry	6.1	C1
2	Limited inquiry	6.2	C1
3	Name discovery	6.3	O
4	Device discovery	6.4	O
5	Bonding	6.5	O
C1: If initiation of bonding is supported, support for at least one inquiry procedure is mandatory, otherwise optional. (Note: support for LMP-pairing is mandatory [2] .)			

6.1 GENERAL INQUIRY

6.1.1 Purpose

The purpose of the general inquiry procedure is to provide the initiator with the Bluetooth device address, clock, Class of Device and used page scan mode of general discoverable devices (i.e. devices that are in range with regard to the initiator and are set to scan for inquiry messages with the General Inquiry Access Code). Also devices in limited discoverable mode will be discovered using general inquiry.

The general inquiry should be used by devices that need to discover devices that are made discoverable continuously or for no specific condition.

6.1.2 Term on UI level

'Bluetooth Device Inquiry'.

6.1.3 Description

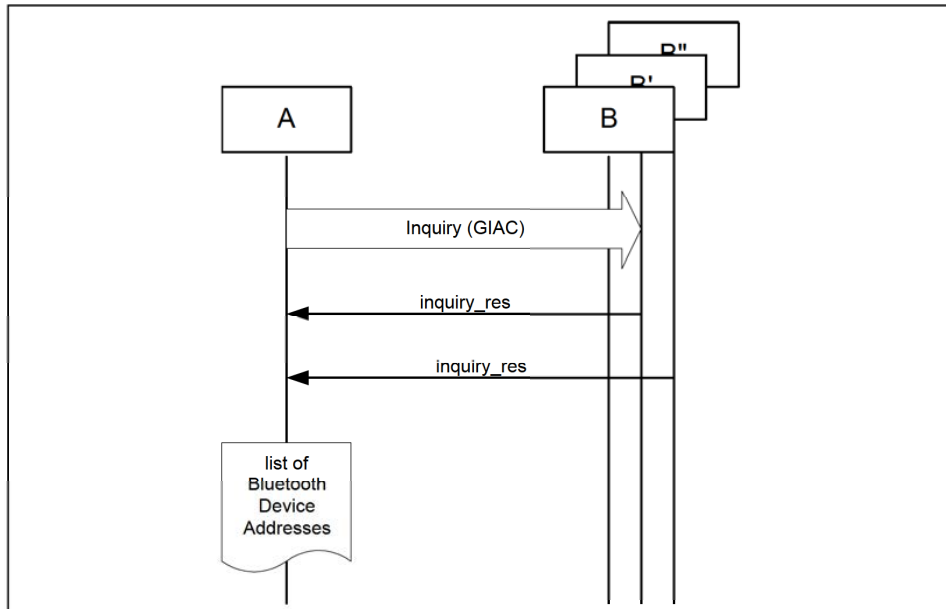


Figure 6.1: General inquiry, where B is a device in non-discoverable mode, B' is a device in limited discoverable mode and B'' is a device in general discoverable mode. (Note that all discoverable devices are discovered using general inquiry, independent of which discoverable mode they are in.)

6.1.4 Conditions

When general inquiry is initiated by a Bluetooth device, the INQUIRY state shall last $T_{\text{GAP}}(100)$ or longer, unless the inquirer collects enough responses and determines to abort the INQUIRY state earlier. The Bluetooth device shall perform inquiry using the GIAC.

In order to receive inquiry response, the remote devices in range have to be made discoverable (limited or general).

6.2 LIMITED INQUIRY

6.2.1 Purpose

The purpose of the limited inquiry procedure is to provide the initiator with the Bluetooth device address, clock, Class of Device and used page scan mode of limited discoverable devices. The latter devices are devices that are in range with regard to the initiator, and may be set to scan for inquiry messages with the Limited Inquiry Access Code, in addition to scanning for inquiry messages with the General Inquiry Access Code.

The limited inquiry should be used by devices that need to discover devices that are made discoverable only for a limited period of time, during temporary



conditions or for a specific event. Since it is not guaranteed that the discoverable device scans for the LIAC, the initiating device may choose any inquiry procedure (general or limited). Even if the remote device that is to be discovered is expected to be made limited discoverable (e.g. when a dedicated bonding is to be performed), the limited inquiry should be done in sequence with a general inquiry in such a way that both inquiries are completed within the time the remote device is limited discoverable, i.e. at least $T_{\text{GAP}}(103)$.

6.2.2 Term on UI level

'Bluetooth Device Inquiry'.

6.2.3 Description

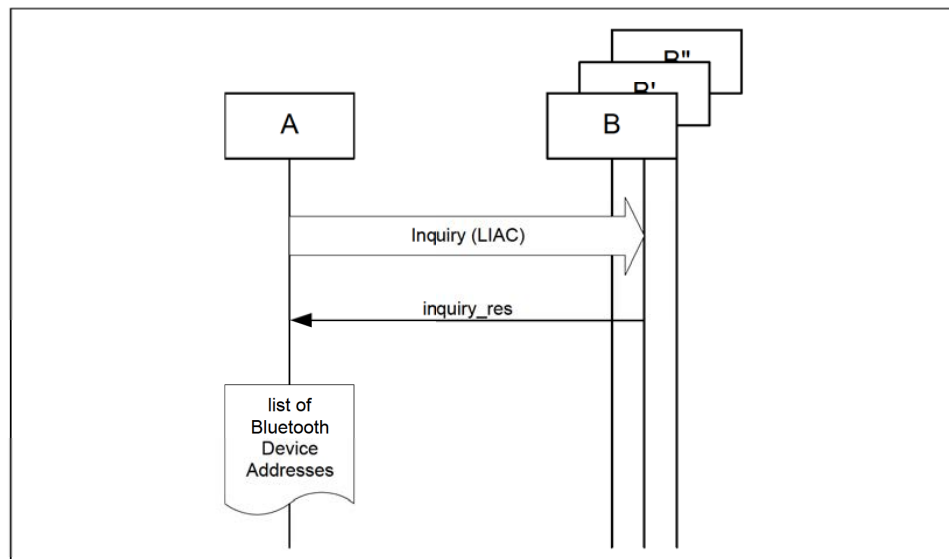


Figure 6.2: Limited inquiry where B is a device in non-discoverable mode, B' is a device in limited discoverable mode and B'' is a device in general discoverable mode. (Note that only limited discoverable devices can be discovered using limited inquiry.)

6.2.4 Conditions

When limited inquiry is initiated by a Bluetooth device, the INQUIRY state shall last $T_{\text{GAP}}(100)$ or longer, unless the inquirer collects enough responses and determines to abort the INQUIRY state earlier. The Bluetooth device shall perform inquiry using the LIAC.

In order to receive inquiry response, the remote devices in range has to be made limited discoverable.



6.3 NAME DISCOVERY

6.3.1 Purpose

The purpose of name discovery is to provide the initiator with the Bluetooth Device Name of connectable devices (i.e. devices in range that will respond to paging).

6.3.2 Term on UI level

'Bluetooth Device Name Discovery'.

6.3.3 Description

6.3.3.1 Name request

Name request is the procedure for retrieving the Bluetooth Device Name from a connectable Bluetooth device. It is not necessary to perform the full link establishment procedure (see [Section 7.1 on page 209](#)) in order to just to get the name of another device.

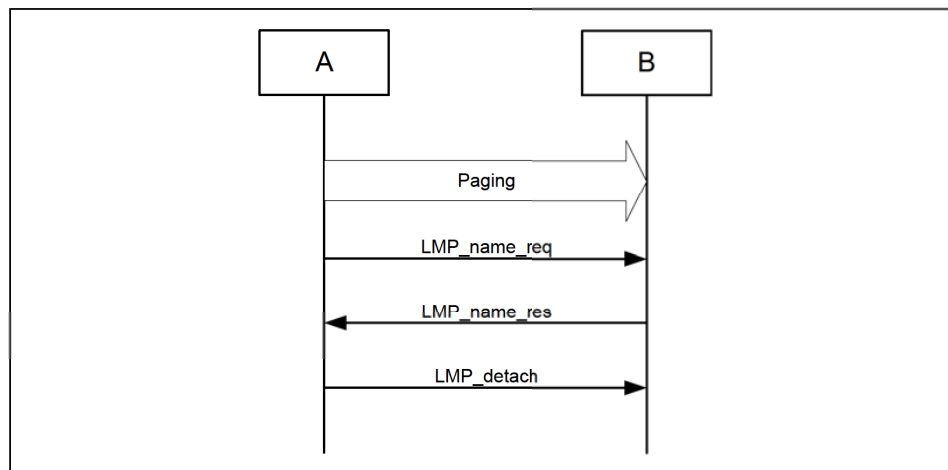


Figure 6.3: Name request procedure.

6.3.3.2 Name discovery

Name discovery is the procedure for retrieving the Bluetooth Device Name from connectable Bluetooth devices by performing name request towards known devices (i.e. Bluetooth devices for which the Bluetooth Device Addresses are available).

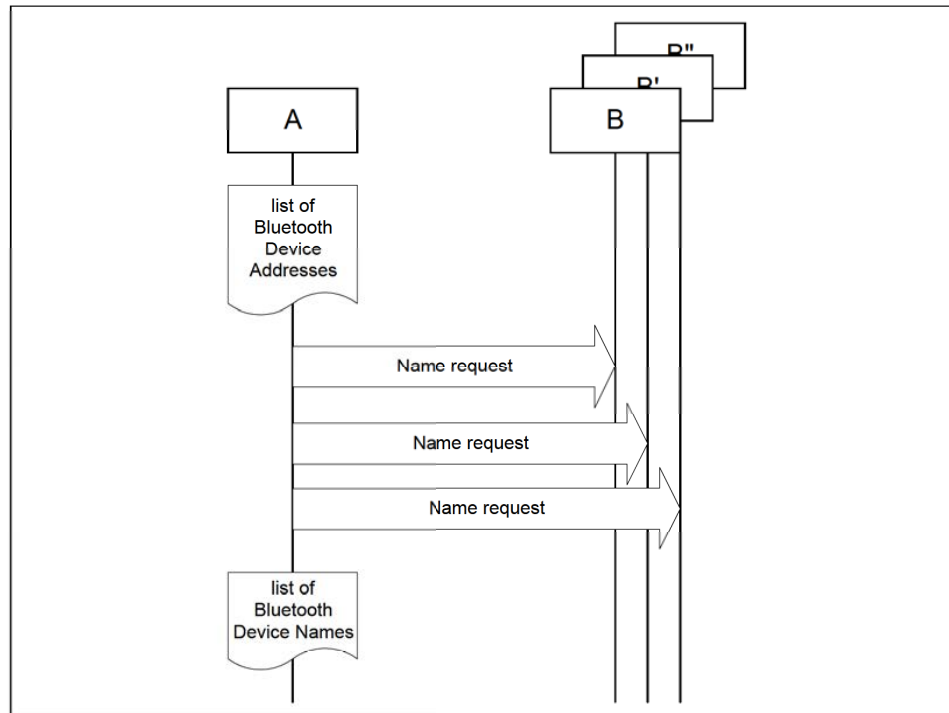


Figure 6.4: Name discovery procedure.

6.3.4 Conditions

In the name request procedure, the initiator will use the Device Access Code of the remote device as retrieved immediately beforehand – normally through an inquiry procedure.

6.4 DEVICE DISCOVERY

6.4.1 Purpose

The purpose of device discovery is to provide the initiator with the Bluetooth Device Address, clock, Class of Device, used page scan mode and Bluetooth device name of discoverable devices.

6.4.2 Term on UI level

'Bluetooth Device Discovery'.

6.4.3 Description

During the device discovery procedure, first an inquiry (either general or limited) is performed, and then name discovery is done towards some or all of the devices that responded to the inquiry.

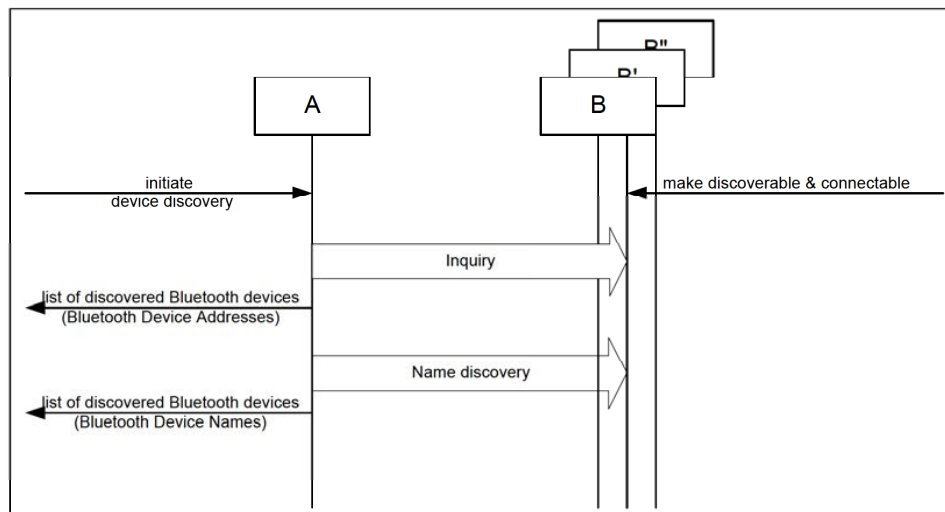


Figure 6.5: Device discovery procedure.

6.4.4 Conditions

Conditions for both inquiry (general or limited) and name discovery must be fulfilled (i.e. devices discovered during device discovery must be both discoverable and connectable).

6.5 BONDING

6.5.1 Purpose

The purpose of bonding is to create a relation between two Bluetooth devices based on a common link key (a bond). The link key is created and exchanged (pairing) during the bonding procedure and is expected to be stored by both Bluetooth devices, to be used for future authentication.

In addition to pairing, the bonding procedure can involve higher layer initialization procedures.

6.5.2 Term on UI level

'Bluetooth Bonding'



6.5.3 Description

Two aspects of the bonding procedure are described here. Dedicated bonding is what is done when the two devices are explicitly set to perform only a creation and exchange of a common link key.

General bonding is included to indicate that the framework for the dedicated bonding procedure is the same as found in the normal channel and connection establishment procedures. This means that pairing may be performed successfully if A has initiated bonding while B is in its normal connectable and security modes.

The main difference with bonding, as compared to a pairing done during link or channel establishment, is that for bonding it is the paging device (A) that must initiate the authentication.

6.5.3.1 General bonding

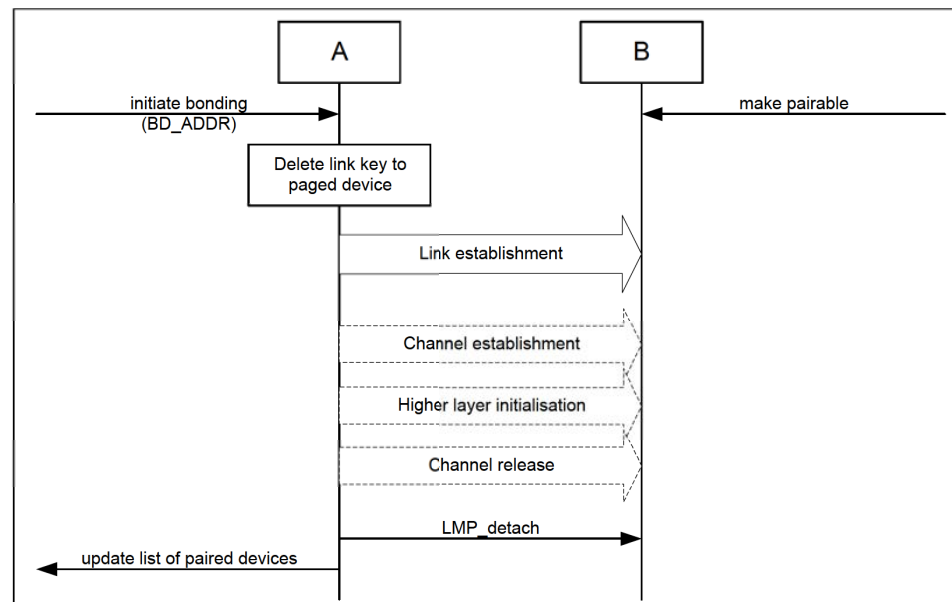


Figure 6.6: General description of bonding as being the link establishment procedure executed under specific conditions on both devices, followed by an optional higher layer initialization process.

6.5.3.2 Dedicated bonding

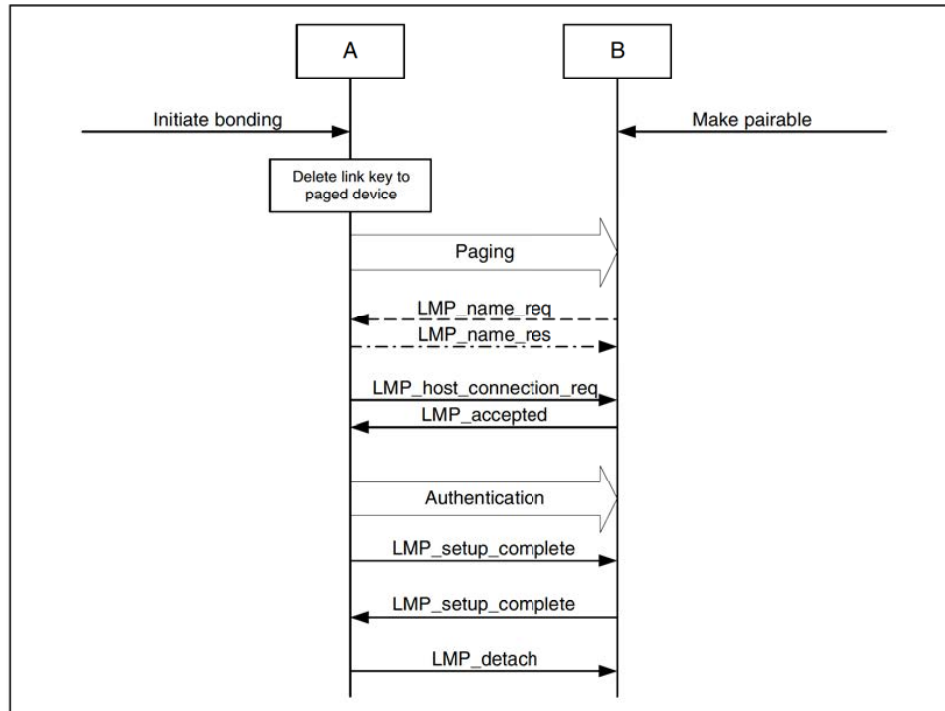


Figure 6.7: Bonding as performed when the purpose of the procedure is only to create and exchange a link key between two Bluetooth devices.

6.5.4 Conditions

Before bonding can be initiated, the initiating device (A) must know the Device Access Code of the device to pair with. This is normally done by first performing device discovery. A Bluetooth Device that can initiate bonding (A) should use limited inquiry, and a Bluetooth Device that accepts bonding (B) should support the limited discoverable mode.

Bonding is in principle the same as link establishment with the conditions:

- The paged device (B) shall be set into pairable mode. The paging device (A) is assumed to allow pairing since it has initiated the bonding procedure.
- The paging device (the initiator of the bonding procedure, A) shall initiate authentication.
- Before initiating the authentication part of the bonding procedure, the paging device should delete any link key corresponding to a previous bonding with the paged device.

7 ESTABLISHMENT PROCEDURES

	Procedure	Ref.	Support in A	Support in B
1	Link establishment	7.1	M	M
2	Channel establishment	7.2	O	M
3	Connection establishment	7.3	O	O

Table 7.1: Establishment procedures

The establishment procedures defined here do not include any discovery part. Before establishment procedures are initiated, the information provided during device discovery (in the FHS packet of the inquiry response or in the response to a name request) has to be available in the initiating device. This information is:

- The Bluetooth Device Address (BD_ADDR) from which the Device Access Code is generated;
- The system clock of the remote device;
- The page scan mode used by the remote device.

Additional information provided during device discovery that is useful for making the decision to initiate an establishment procedure is:

- The Class of device;
- The Device name.

7.1 LINK ESTABLISHMENT

7.1.1 Purpose

The purpose of the link establishment procedure is to establish a physical link (of ACL type) between two Bluetooth devices using procedures from [\[1\]](#) and [\[2\]](#).

7.1.2 Term on UI level

'Bluetooth link establishment'



7.1.3 Description

In this sub-section, the paging device (A) is in security mode 3. The paging device cannot during link establishment distinguish if the paged device (B) is in security mode 1 or 2.

7.1.3.1 B in security mode 1 or 2

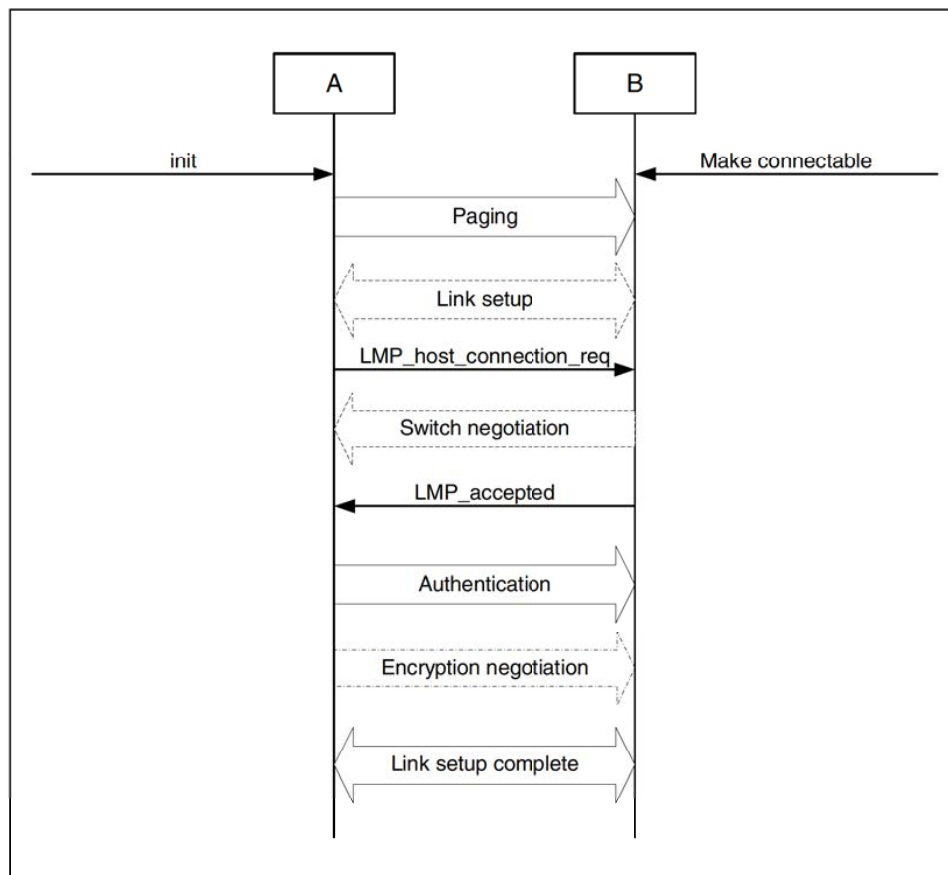


Figure 7.1: Link establishment procedure when the paging device (A) is in security mode 3 and the paged device (B) is in security mode 1 or 2.

7.1.3.2 B in security mode 3

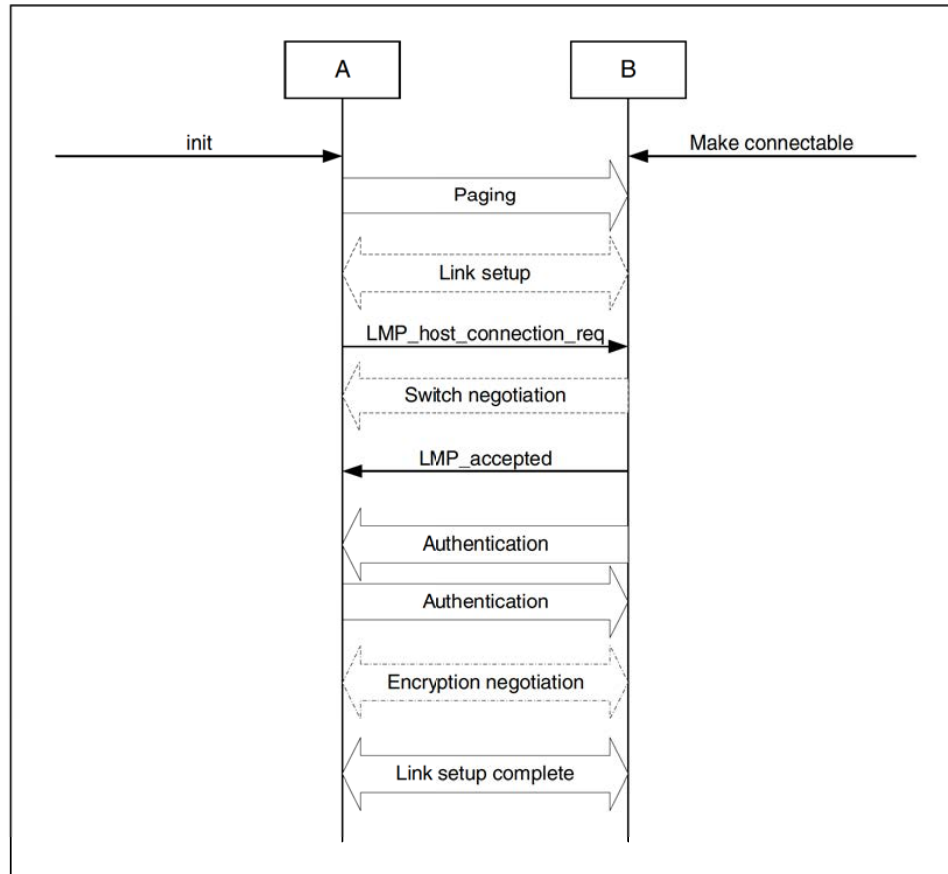


Figure 7.2: Link establishment procedure when both the paging device (A) and the paged device (B) are in security mode 3.

7.1.4 Conditions

The paging procedure shall be according to [1] and the paging device should use the Device access code and page mode received through a previous inquiry. When paging is completed, a physical link between the two Bluetooth devices is established.

If role switching is needed (normally it is the paged device that has an interest in changing the master/slave roles) it should be done as early as possible after the physical link is established. If the paging device does not accept the switch, the paged device has to consider whether to keep the physical link or not.

Both devices may perform link setup (using LMP procedures that require no interaction with the host on the remote side). Optional LMP features can be used after having confirmed (using LMP_feature_req) that the other device supports the feature.



When the paging device needs to go beyond the link setup phase, it issues a request to be connected to the host of the remote device. If the paged device is in security mode 3, this is the trigger for initiating authentication.

The paging device shall send LMP_host_connection_req during link establishment (i.e. before channel establishment) and may initiate authentication only after having sent LMP_host_connection_request.

After an authentication has been performed, any of the devices can initiate encryption.

Further link configuration may take place after the LMP_host_connection_req. When both devices are satisfied, they send LMP_setup_complete.

Link establishment is completed when both devices have sent LMP_setup_complete.

7.2 CHANNEL ESTABLISHMENT

7.2.1 Purpose

The purpose of the channel establishment procedure is to establish a Bluetooth channel (L2CAP channel) between two Bluetooth devices using [3].

7.2.2 Term on UI level

'Bluetooth channel establishment'.

7.2.3 Description

In this sub-section, the initiator (A) is in security mode 3. During channel establishment, the initiator cannot distinguish if the acceptor (B) is in security mode 1 or 3.

7.2.3.1 B in security mode 2

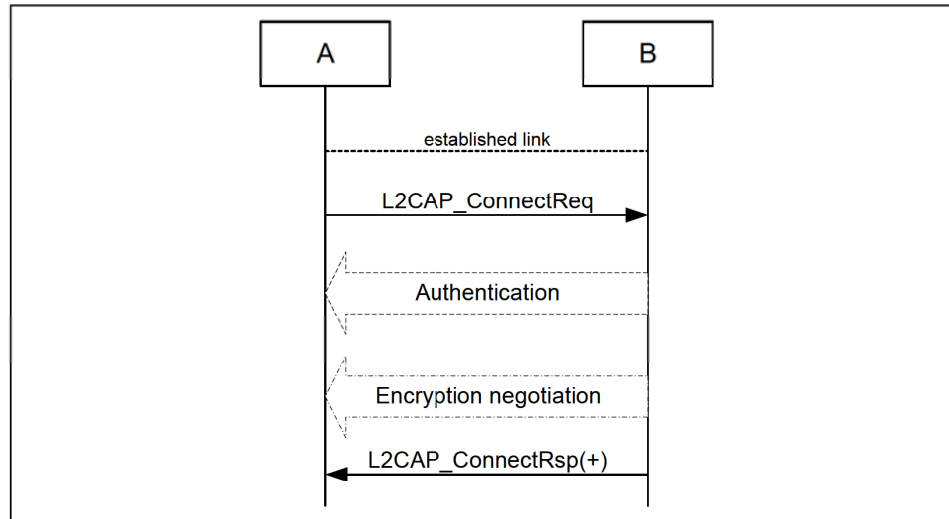


Figure 7.3: Channel establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 2.

7.2.3.2 B in security mode 1 or 3

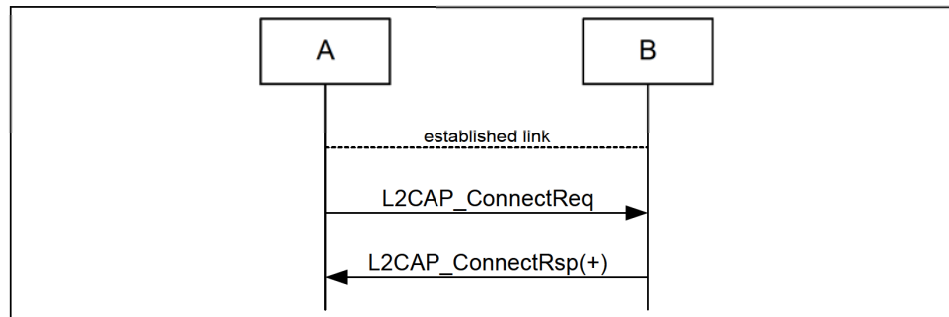


Figure 7.4: Channel establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 1 or 3.

7.2.4 Conditions

Channel establishment starts after link establishment is completed when the initiator sends a channel establishment request (L2CAP_ConnectReq).

Depending on security mode, security procedures may take place after the channel establishment has been initiated.

Channel establishment is completed when the acceptor responds to the channel establishment request (with a positive L2CAP_ConnectRsp).



7.3 CONNECTION ESTABLISHMENT

7.3.1 Purpose

The purpose of the connection establishment procedure is to establish a connection between applications on two Bluetooth devices.

7.3.2 Term on UI level

'Bluetooth connection establishment'

7.3.3 Description

In this sub-section, the initiator (A) is in security mode 3. During connection establishment, the initiator cannot distinguish if the acceptor (B) is in security mode 1 or 3.

7.3.3.1 B in security mode 2

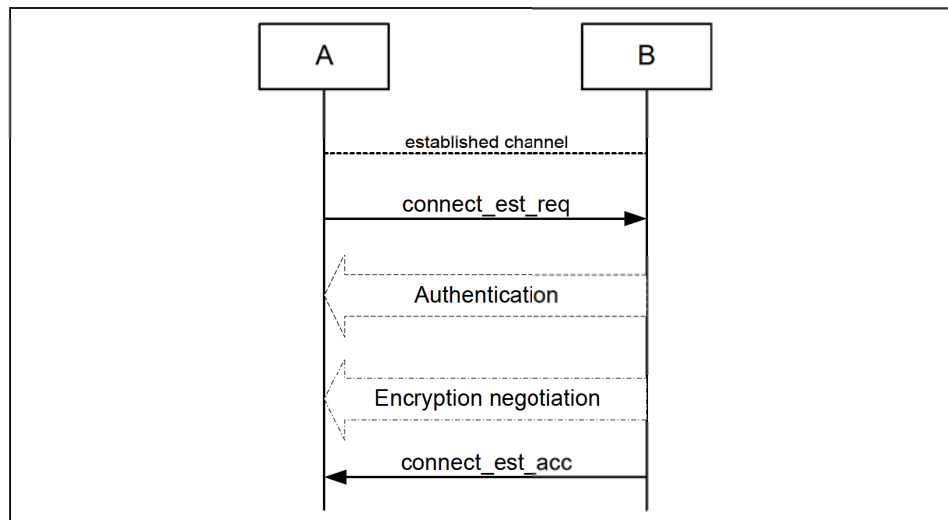


Figure 7.5: Connection establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 2.

7.3.3.2 B in security mode 1 or 3

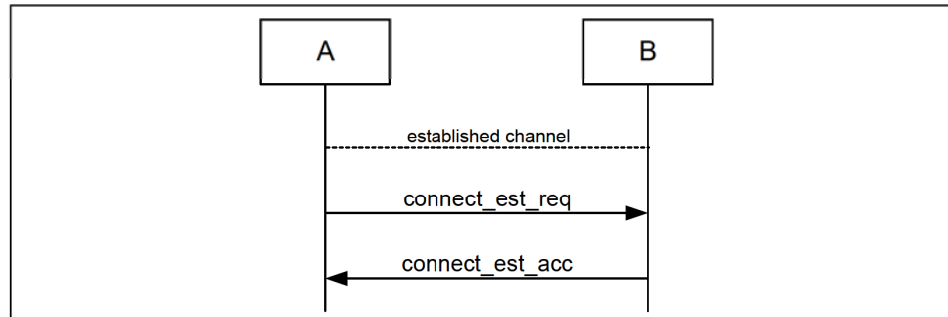


Figure 7.6: Connection establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 1 or 3.

7.3.4 Conditions

Connection establishment starts after channel establishment is completed, when the initiator sends a connection establishment request ('connect_est_req' is application protocol-dependent). This request may be a TCS SETUP message [5] in the case of a Bluetooth telephony application [Cordless Telephony Profile](#), or initialization of RFCOMM and establishment of DLC [4] in the case of a serial port-based application [Serial Port Profile](#) (although neither TCS or RFCOMM use the term 'connection' for this).

Connection establishment is completed when the acceptor accepts the connection establishment request ('connect_est_acc' is application protocol dependent).

7.4 ESTABLISHMENT OF ADDITIONAL CONNECTION

When a Bluetooth device has established one connection with another Bluetooth device, it may be available for establishment of:

- A second connection on the same channel, and/or
- A second channel on the same logical link, and/or
- A second physical link.

If the new establishment procedure is to be towards the same device, the security part of the establishment depends on the security modes used. If the new establishment procedure is to be towards a new remote device, the device should behave according to active modes independent of the fact that it already has another physical link established (unless allowed co-incident radio and baseband events have to be handled).





8 DEFINITIONS

In the following, terms written with capital letters refer to states.

8.1 GENERAL DEFINITIONS

Mode . A set of directives that defines how a device will respond to certain events.

Idle . As seen from a remote device, a Bluetooth device is idle, or is in idle mode, when there is no link established between them.

Bond . A relation between two Bluetooth devices defined by creating, exchanging and storing a common link key. The bond is created through the bonding or LMP-pairing procedures.

8.2 CONNECTION-RELATED DEFINITIONS

Physical channel . A synchronized Bluetooth baseband-compliant RF hopping sequence.

Piconet . A set of Bluetooth devices sharing the same physical channel defined by the master parameters (clock and BD_ADDR).

Physical link . A Baseband-level connection¹ between two devices established using paging. A physical link comprises a sequence of transmission slots on a physical channel alternating between master and slave transmission slots.

ACL link . An asynchronous (packet-switched) connection¹ between two devices created on LMP level. Traffic on an ACL link uses ACL packets to be transmitted.

SCO link . A synchronous (circuit-switched) connection¹ for reserved bandwidth communications; e.g. voice between two devices, created on the LMP level by reserving slots periodically on a physical channel. Traffic on an SCO link uses SCO packets to be transmitted. SCO links can be established only after an ACL link has first been established.

Link . Shorthand for an ACL link.

PAGE . A baseband state where a device transmits page trains, and processes any eventual responses to the page trains.

PAGE_SCAN . A baseband state where a device listens for page trains.

1. The term 'connection' used here is not identical to the definition below. It is used in the absence of a more concise term.



Page . The transmission by a device of page trains containing the Device Access Code of the device to which the physical link is requested.

Page scan . The listening by a device for page trains containing its own Device Access Code.

Channel . A logical connection on L2CAP level between two devices serving a single application or higher layer protocol.

Connection . A connection between two peer applications or higher layer protocols mapped onto a channel.

Connecting . A phase in the communication between devices when a connection between them is being established. (Connecting phase follows after the link establishment phase is completed.)

Connect (to service) . The establishment of a connection to a service. If not already done, this includes establishment of a physical link, link and channel as well.

8.3 DEVICE-RELATED DEFINITIONS

Discoverable device . A Bluetooth device in range that will respond to an inquiry (normally in addition to responding to page).

Silent device . A Bluetooth device appears as silent to a remote device if it does not respond to inquiries made by the remote device. A device may be silent due to being non-discoverable or due to baseband congestion while being discoverable.

Connectable device . A Bluetooth device in range that will respond to a page.

Trusted device . A paired device that is explicitly marked as trusted.

Paired device . A Bluetooth device with which a link key has been exchanged (either before connection establishment was requested or during connecting phase).

Pre-paired device . A Bluetooth device with which a link key was exchanged, and the link key is stored, before link establishment.

Un-paired device . A Bluetooth device for which there was no exchanged link key available before connection establishment was request.

Known device . A Bluetooth device for which at least the BD_ADDR is stored.

Un-known device . A Bluetooth device for which no information (BD_ADDR, link key or other) is stored.

Authenticated device . A Bluetooth device whose identity has been verified during the lifetime of the current link, based on the authentication procedure.



8.4 PROCEDURE-RELATED DEFINITIONS

Paging . A procedure for establishing a physical link of ACL type on baseband level, consisting of a page action of the initiator and a page scan action of the responding device.

Link establishment . A procedure for establishing a link on LMP level. A link is established when both devices have agreed that LMP setup is completed.

Channel establishment . A procedure for establishing a channel on L2CAP level.

Connection establishment . A procedure for creating a connection mapped onto a channel.

Creation of a trusted relationship . A procedure where the remote device is marked as a trusted device. This includes storing a common link key for future authentication and pairing (if the link key is not available).

Creation of a secure connection . A procedure of establishing a connection, including authentication and encryption.

Device discovery . A procedure for retrieving the Bluetooth device address, clock, class-of-device field and used page scan mode from discoverable devices.

Name discovery . A procedure for retrieving the user-friendly name (the Bluetooth device name) of a connectable device.

Service discovery . Procedures for querying and browsing for services offered by or through another Bluetooth device.

8.5 SECURITY-RELATED DEFINITIONS

Authentication . A generic procedure based on LMP-authentication if a link key exists or on LMP-pairing if no link key exists.

LMP-authentication . An LMP level procedure for verifying the identity of a remote device. The procedure is based on a challenge-response mechanism using a random number, a secret key and the BD_ADDR of the non-initiating device. The secret key used can be a previously exchanged link key.

Authorization . A procedure where a user of a Bluetooth device grants a specific (remote) Bluetooth device access to a specific service. Authorization implies that the identity of the remote device can be verified through authentication.

Authorize . The act of granting a specific Bluetooth device access to a specific service. It may be based upon user confirmation, or given the existence of a trusted relationship.



LMP-pairing . A procedure that authenticates two devices, based on a PIN, and subsequently creates a common link key that can be used as a basis for a trusted relationship or a (single) secure connection. The procedure consists of the steps: creation of an initialization key (based on a random number and a PIN), creation and exchange of a common link key and LMP-authentication based on the common link key.

Bonding . A dedicated procedure for performing the first authentication, where a common link key is created and stored for future use.

Trusting . The marking of a paired device as trusted. Trust marking can be done by the user, or automatically by the device (e.g. when in pairable mode) after a successful pairing.



9 APPENDIX A (NORMATIVE): TIMERS AND CONSTANTS

The following timers are required by this profile.

Timer name	Recommended value	Description	Comment
$T_{\text{GAP}}(100)$	10.24 s	Normal time span that a Bluetooth device performs inquiry.	Used during inquiry and device discovery.
$T_{\text{GAP}}(101)$	10.625 ms	Minimum time in INQUIRY_SCAN.	A discoverable Bluetooth device enters INQUIRY_SCAN for at least $T_{\text{GAP}}(101)$ every $T_{\text{GAP}}(102)$.
$T_{\text{GAP}}(102)$	2.56 s	Maximum time between repeated INQUIRY_SCAN enterings.	Maximum value of the inquiry scan interval, $T_{\text{inquiry scan}}$.
$T_{\text{GAP}}(103)$	30.72 s	A Bluetooth device shall not be in a discoverable mode less than $T_{\text{GAP}}(103)$.	Minimum time to be discoverable.
$T_{\text{GAP}}(104)$	1 min.	A Bluetooth device should not be in limited discoverable mode more than $T_{\text{GAP}}(104)$.	Recommended upper limit.
$T_{\text{GAP}}(105)$	100ms	Maximum time between INQUIRY_SCAN enterings	Recommended value
$T_{\text{GAP}}(106)$	100ms	Maximum time between PAGE_SCAN enterings	Recommended value
$T_{\text{GAP}}(107)$	1.28s	Maximum time between PAGE_SCAN enterings (R1 page scan)	Recommended value
$T_{\text{GAP}}(108)$	2.56s	Maximum time between PAGE_SCAN enterings (R2 page scan)	Recommended value

Table 9.1: Defined GAP timers



10 APPENDIX B (INFORMATIVE): INFORMATION FLOWS OF RELATED PROCEDURES

10.1 LMP-AUTHENTICATION

The specification of authentication on link level is found in [2].

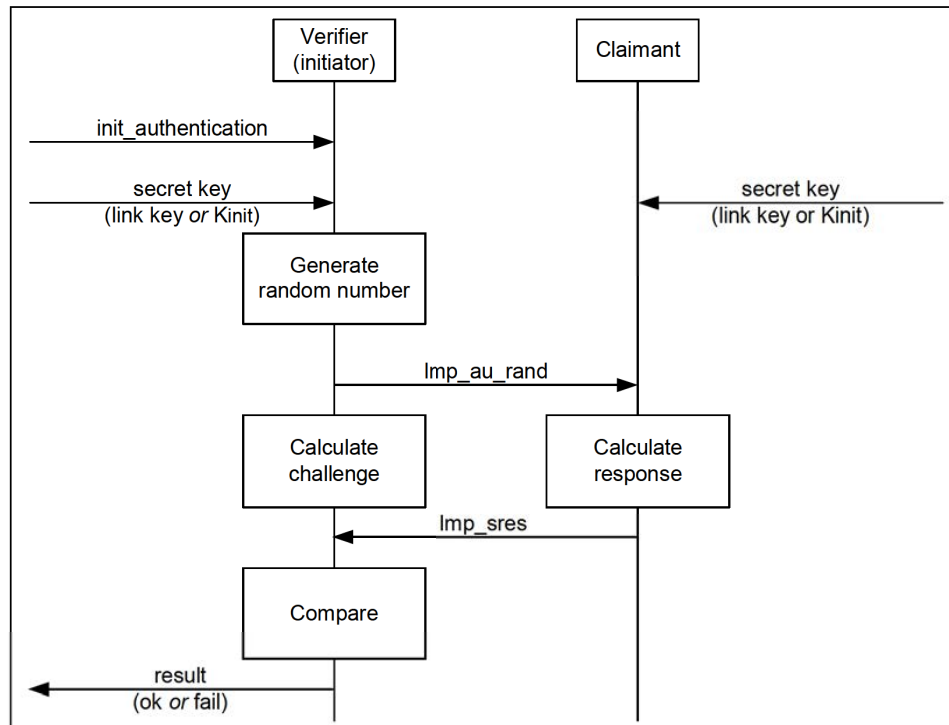


Figure 10.1: LMP-authentication as defined by [2].

The secret key used here is an already exchanged link key.

10.2 LMP-PAIRING

The specification of pairing on link level is found in [2].

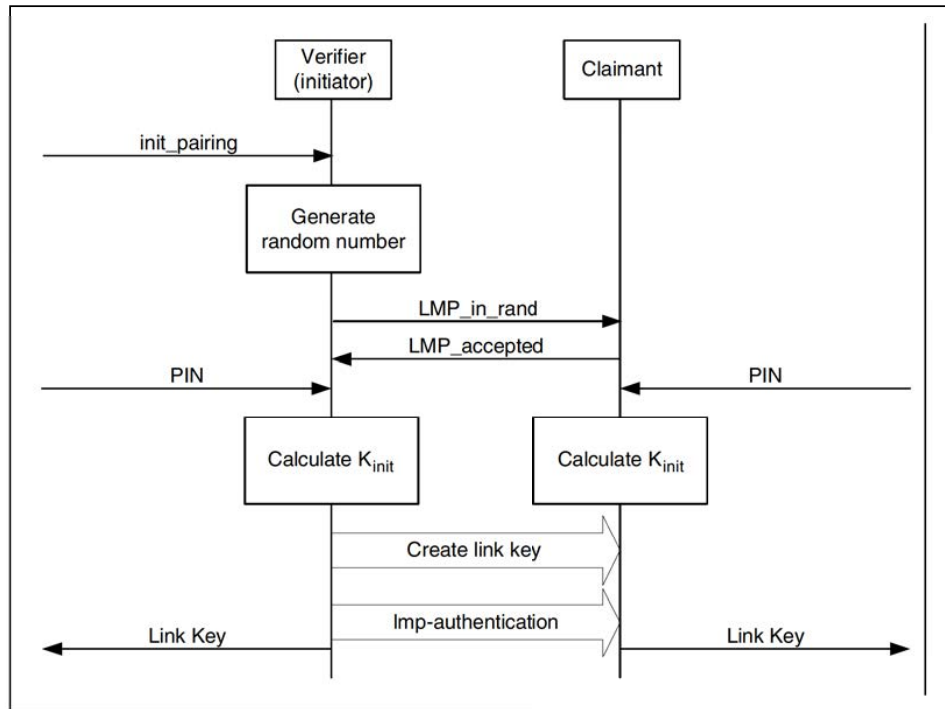


Figure 10.2: LMP-pairing as defined in [2].

The PIN used here is PN_{BB} .

The create link key procedure is described in [Vol. 3, Part C] Section 4.2.2.4 on page 253 and [Vol. 3, Part H] Section 3.2 on page 779. In case the link key is based on a combination key, a mutual authentication takes place and shall be performed irrespective of current security mode.

10.3 SERVICE DISCOVERY

The Service Discovery Protocol [6] specifies what PDUs are used over-the-air to inquire about services and service attributes. The procedures for discovery of supported services and capabilities using the Service Discovery Protocol are described in the [Service Discovery Application Profile](#). This is just an example.

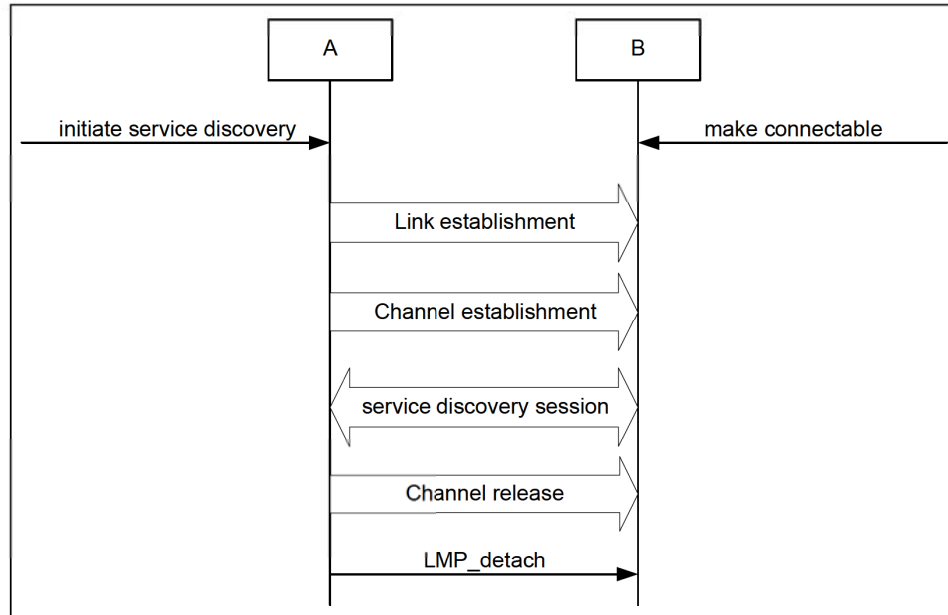


Figure 10.3: Service discovery procedure.





11 REFERENCES

- [1] Bluetooth Baseband Specification
- [2] Bluetooth Link Manager Protocol
- [3] Bluetooth Logical Link Control and Adaptation Protocol
- [4] Bluetooth RFCOMM
- [5] Bluetooth Telephony Control Specification
- [6] Bluetooth Service Discovery Protocol
- [7] Bluetooth Service Discovery Application Profile
- [8] Bluetooth Cordless Telephony Profile
- [9] Bluetooth Serial Port Profile
- [10] Bluetooth Security Architecture (white paper)
- [11] Bluetooth Assigned Numbers
https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers



Core System Package [Host volume]
Part D

TEST SUPPORT



CONTENTS

11	Test Methodology	231
1.1	Test Scenarios.....	231
1.1.1	Test setup	231
1.1.2	Transmitter Test.....	232
1.1.2.1	Packet Format.....	233
1.1.2.2	Pseudorandom Sequence	234
1.1.2.3	Control of Transmit Parameters	235
1.1.2.4	Power Control	235
1.1.2.5	Switch Between Different Frequency Settings	235
1.1.2.6	Adaptive Frequency Hopping.....	236
1.1.3	LoopBack test.....	237
1.1.4	Pause test	240
1.2	References.....	240
2	Test Control Interface (TCI)	241
2.1	Introduction	241
2.1.1	Terms used.....	241
2.1.2	Usage of the interface	241
2.2	TCI Configurations	242
2.2.1	Bluetooth RF requirements.....	242
2.2.1.1	Required interfaces.....	242
2.2.2	Bluetooth protocol requirements.....	243
2.2.2.1	Required interfaces.....	243
2.2.3	Bluetooth profile requirements.....	244
2.2.3.1	Required interfaces.....	244
2.3	TCI Configuration and Usage.....	245
2.3.1	Transport layers.....	245
2.3.1.1	Physical bearer	245
2.3.1.2	Software bearer	245
2.3.2	Baseband and link manager qualification.....	246
2.3.3	HCI qualification	248

1 TEST METHODOLOGY

This section describes the test mode for hardware and low-level functionality tests of Bluetooth devices. The test mode includes transmitter tests (packets with constant bit patterns) and loop back tests.

The test mode supports testing of the Bluetooth transmitter and receiver. It is intended mainly for certification/compliance testing of the radio and baseband layer, and may also be used for regulatory approval or in-production and after-sales testing.

1.1 TEST SCENARIOS

A device in test mode shall not support normal operation. For security reasons the test mode is designed such that it offers no benefit to the user. Therefore, no data output or acceptance on a HW or SW interface shall be allowed.

1.1.1 Test setup

The setup consists of a device under test (DUT) and a tester. Optionally, additional measurement equipment may be used.

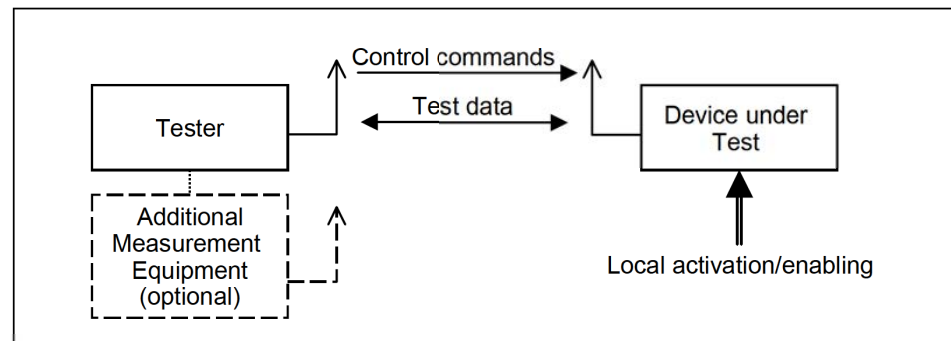


Figure 1.1: Setup for Test Mode

Tester and DUT form a piconet where the tester acts as master and has full control over the test procedure. The DUT acts as slave.

The control is done via the air interface using LMP commands (see [\[Vol. 3, Part C\] Section 4.7.3 on page 291](#)). Hardware interfaces to the DUT may exist, but are not subject to standardization.

The test mode is a special state of the Bluetooth model. For security and type approval reasons, a Bluetooth device in test mode shall not support normal operation. When the DUT leaves the test mode it enters the standby state. After power-off the Bluetooth device shall return to standby state.



1.1.2 Transmitter Test

The Bluetooth device transmits a constant bit pattern. This pattern is transmitted periodically with packets aligned to the slave TX timing of the piconet formed by tester and DUT. The same test packet is repeated for each transmission.

The transmitter test is started when the master sends the first POLL packet. In non-hopping mode agreed frequency is used for this POLL packet.

The tester (master) transmits control or POLL packets in the master-to-slave transmission slots. The DUT (slave) shall transmit packets in the following slave-to-master transmission slot. The tester's polling interval is fixed and defined by the LMP_test_control PDU. The device under test may transmit its burst according to the normal timing even if no packet from the tester was received. In this case, the ARQN bit is shall be set to NAK.

The burst length may exceed the length of a one slot packet. In this case the tester may take the next free master TX slot for polling. The timing is illustrated in [Figure 1.2](#).

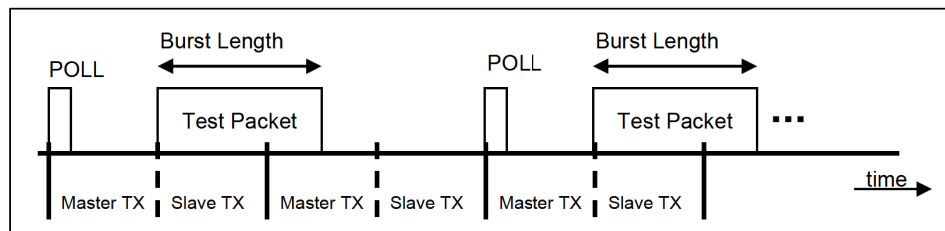


Figure 1.2: Timing for Transmitter Test

1.1.2.1 Packet Format

The test packet is a normal Bluetooth packet, see [Figure 1.3](#). For the payload itself see below.

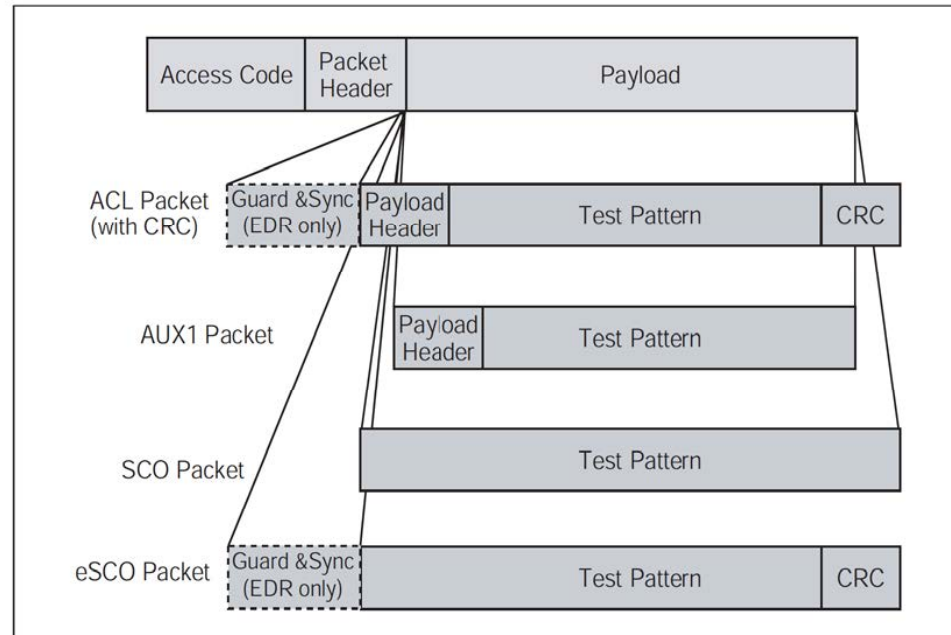


Figure 1.3: General Format of TX Packet

During configuration the tester defines:

- the packet type to be used
- payload length

For the payload length, the restrictions from the baseband specification shall apply (see [“Baseband Specification” on page 55\[vol. 3\]](#)). In case of ACL, SCO and eSCO packets the payload structure defined in the baseband specification is preserved as well, see [Figure 1.3 on page 233](#).

For the transmitter test mode, only packets without FEC should be used; i.e. HV3, EV3, EV5, DH1, DH3, DH5, 2-EV3, 2-EV5, 3-EV3, 3-EV5, 2-DH1, 2-DH3, 2-DH5, 3-DH1, 3-DH3, 3-DH5 and AUX1 packets.

In transmitter test mode, the packets exchanged between the tester and the DUT shall not be scrambled with the whitening sequence. Whitening shall be turned off when the DUT has accepted to enter the transmitter test mode, and shall be turned on when the DUT has accepted to exit the transmitter test mode, see [Figure 1.4 on page 234](#). Implementations shall insure that retransmissions of the LMP_accepted messages use the same whitening status as used in the original LMP_accepted.

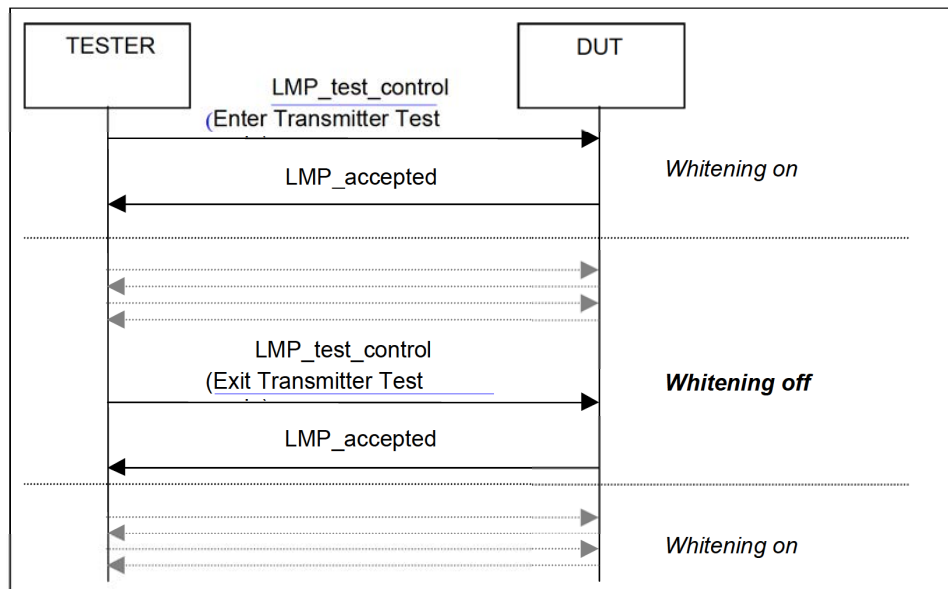


Figure 1.4: Use of whitening in Transmitter mode

1.1.2.2 Pseudorandom Sequence

The same pseudorandom sequence of bits shall be used for each transmission (i.e. the packet is repeated). A PRBS-9 Sequence is used, see [2] and [3].

The properties of this sequence are as follows (see [3]). The sequence may be generated in a nine-stage shift register whose 5th and 9th stage outputs are added in a modulo-two addition stage (see Figure 1.5), and the result is fed back to the input of the first stage. The sequence begins with the first ONE of 9 consecutive ONES; i.e. the shift register is initialized with nine ones.

- Number of shift register stages: 9
- Length of pseudo-random sequence: $2^9 - 1 = 511$ bits
- Longest sequence of zeros: 8 (non-inverted signal)

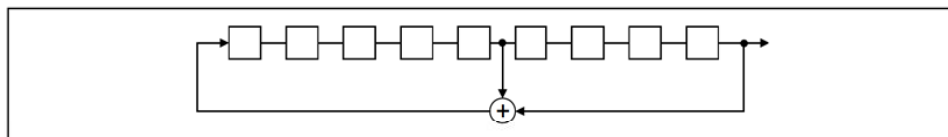


Figure 1.5: Linear Feedback Shift Register for Generation of the PRBS sequence



1.1.2.3 Control of Transmit Parameters

The following parameters can be set to configure the transmitter test:

1. Bit pattern:
 - Constant zero
 - Constant one
 - Alternating 1010...¹
 - Alternating 1111 0000 1111 0000...⁴
 - Pseudorandom bit pattern
 - Transmission off
2. Frequency selection:
 - Single frequency
 - Normal hopping
3. TX frequency
 - $k \Rightarrow f := (2402 + k) \text{ MHz}$
4. Default poll period in TDD frames ($n * 1.25 \text{ ms}$)
5. Packet Type
6. Length of Test Sequence (user data of packet definition in [“Baseband Specification” on page 55\[vol. 3\]](#))

1.1.2.4 Power Control

If adaptive power control is tested, the normal LMP commands will be used. The DUT shall start transmitting at the maximum power and shall reduce/increase its power by one step on every LMP_incr_power_req or LMP_decr_power_req command received.

1.1.2.5 Switch Between Different Frequency Settings

A change in the frequency selection becomes effective when the LMP procedure is completed:

When the tester receives the LMP_accepted it shall then transmit POLL packets containing the Ack for at least 8 slots (4 transmissions). When these transmissions have been completed the tester shall change to the new frequency hop and whitening settings.

After sending LMP_accepted the DUT shall wait for the LC level Ack for the LMP_accepted. When this is received it shall change to the new frequency hop and whitening settings.

1. It is recommended that the sequence starts with a one; but, as this is irrelevant for measurements, it is also allowed to start with a zero.



There will be an implementation defined delay after sending the LMP_accepted before the TX or loopback test starts. Testers shall be able to cope with this.

***Note:** Loss of the LMP_Accepted packet will eventually lead to a loss of frequency synchronization that cannot be recovered. Similar problems occur in normal operation, when the hopping pattern changes.*

1.1.2.6 Adaptive Frequency Hopping

Adaptive Frequency Hopping (AFH) shall only be used when the Hopping Mode is set to 79 channels (e.g. Hopping Mode = 1) in the LMP_test_control PDU. If AFH is used, the normal LMP commands and procedures shall be used. When AFH is enabled prior to entering test mode it shall continue to be used with the same parameters if Hopping Mode = 1 until the AFH parameters are changed by the LMP_set_AFH PDU.

The channel classification reporting state shall be retained upon entering or exiting Test Mode. The DUT shall change the channel classification reporting state in Test Mode based on control messages from the tester (LMP_channel_classification_req) and from the Host (HCI Write_AFH_Channel_Classification_Mode).



1.1.3 LoopBack test

In loopback, the device under test receives normal baseband packets containing payload *Accepted* from the tester. The received packets shall be decoded in the DUT, and the payload shall be sent back using the same packet type. The return packet shall be sent back in either the slave-to-master transmission slot directly following the transmission of the tester, or it is delayed and sent back in the slave-to-master transmission slot after the next transmission of the tester (see [Figure 1.7](#) to [Figure 1.9 on page 239](#)).

There is no signalling to determine or control the mode. The device behavior shall be fixed or adjusted by other means, and shall not change randomly.

The tester can select, whether whitening is on or off. This setting holds for both up- and downlink. For switching the whitening status, the same rules as in [Section 1.1.2 on page 232](#) ([Figure 1.4](#)) shall apply.

The following rules apply (for illustration see [Figure 1.6 on page 238](#)):

- If the synch word was not detected, the DUT shall not reply.
- If the header error check (HEC) fails, the DUT shall either reply with a NULL packet with the ARQN bit set to NAK or send nothing.
- If the packet contains an LMP message relating to the control of the test mode this command shall be executed and the packet shall not be returned, though ACK or NAK shall be returned as per the usual procedure. Other LMP commands are ignored and no packet is returned.
- The payload FEC is decoded and the payload shall be encoded again for transmission. This allows testing of the FEC handling. If the pure bit error rate shall be determined the tester chooses a packet type without FEC.
- The CRC is shall be evaluated. In the case of a failure, ARQN=NAK shall be returned. The payload shall be returned as received.
A new CRC for the return packet shall be calculated for the returned payload regardless of whether the CRC was valid or not.
- If the CRC fails for a packet with a CRC and a payload header, the number of bytes as indicated in the (possibly erroneous) payload header shall be looped back.

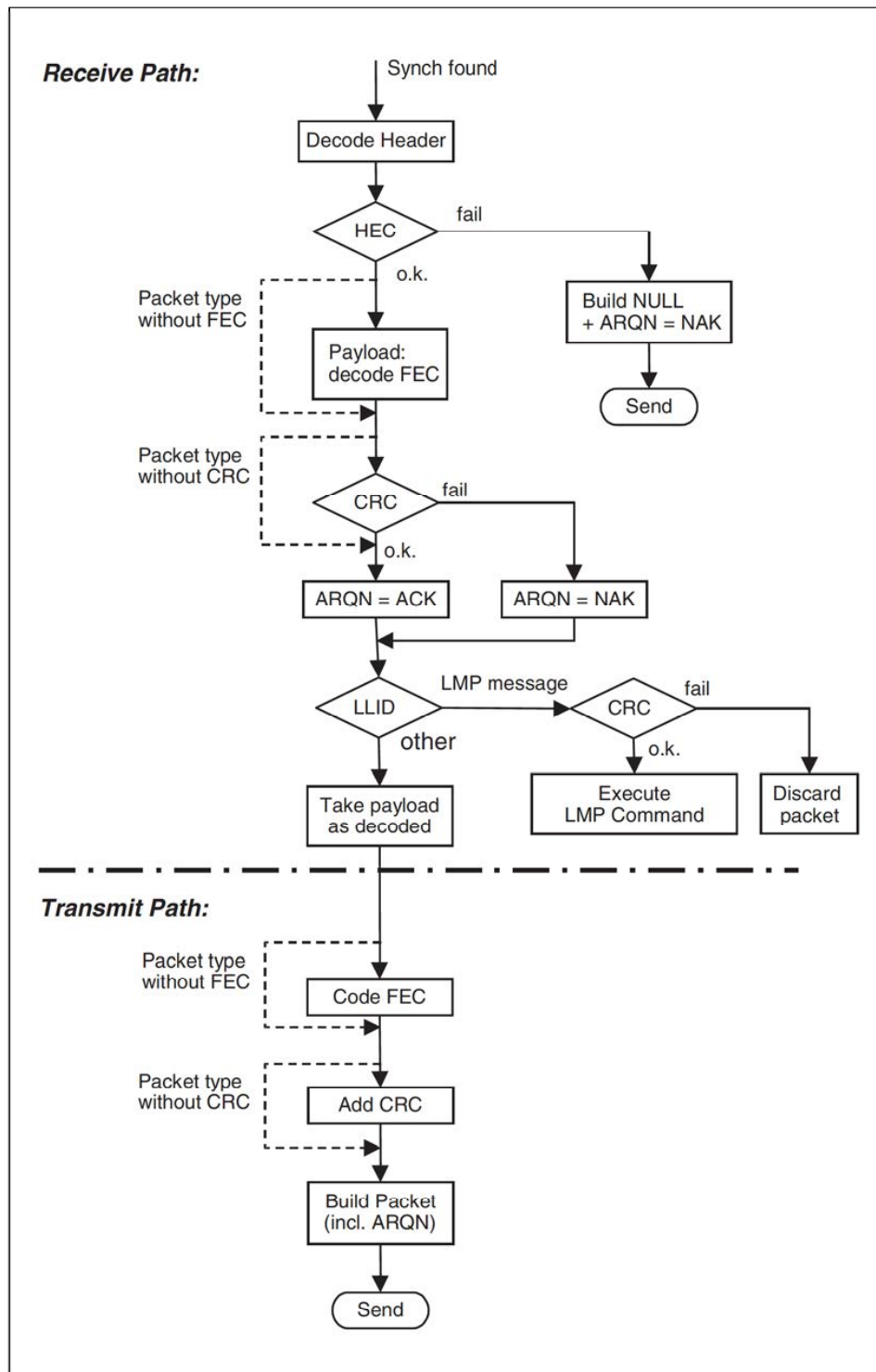


Figure 1.6: DUT Packet Handling in Loop Back Test

The timing for normal and delayed loopback is illustrated in [Figure 1.7](#) to [Figure 1.9](#):

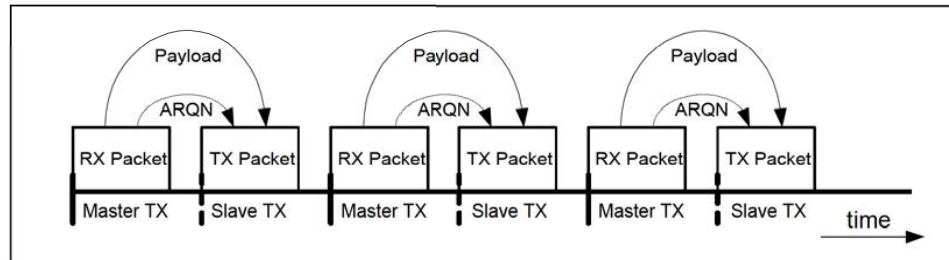


Figure 1.7: Payload & ARQN handling in normal loopback.

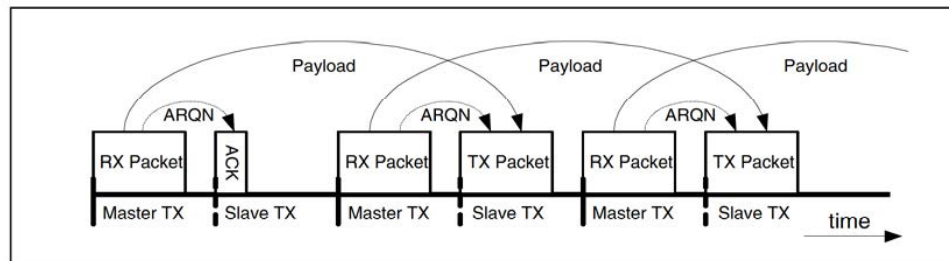


Figure 1.8: Payload & ARQN handling in delayed loopback - start.

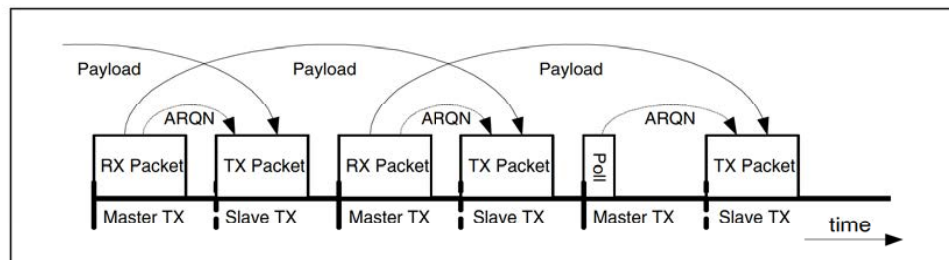


Figure 1.9: Payload & ARQN handling in delayed loopback - end.

The whitening is performed in the same way as it is used in normal active mode.

The following parameters can be set to configure the loop back test:

1. Packet Class¹
 - ACL Packets
 - SCO Packets
 - eSCO Packets
 - ACL Packets without whitening

1. This is included because, in the future, the packet type numbering may not remain unambiguous.



- SCO Packets without whitening
 - eSCO Packets without whitening
2. Frequency Selection
 - Single frequency (independent for RX and TX)
 - Normal hopping
 3. Power level: (To be used according radio specification requirements)
 - power control or fixed TX power

The switch of the frequency setting is done exactly as for the transmitter test (see [Section 1.1.2.5 on page 235](#)).

1.1.4 Pause test

Pause test is used by testers to put the device under test into Pause Test mode from either the loopback or transmitter test modes.

When an LMP_test_control PDU that specifies Pause Test is received the DUT shall stop the current test and enter Pause Test mode. In the case of a transmitter test this means that no more packets shall be transmitted. While in Pause Test mode the DUT shall respond normally to POLL packets (i.e. responds with a NULL packet). The DUT shall also respond normally to all the LMP packets that are allowed in test mode.

When the test scenario is set to Pause Test all the other fields in the LMP_test_control PDU shall be ignored. There shall be no change in hopping scheme or whitening as a result of a request to pause test.

1.2 REFERENCES

- [1] Bluetooth Link Manager Protocol.
- [2] CCITT Recommendation O.153 (1992), Basic parameters for the measurement of error performance at bit rates below the primary rate.
- [3] ITU-T Recommendation O.150 (1996), General requirements for instrumentation for performance measurements on digital transmission equipment.
- [4] Bluetooth Baseband Specification.



2 TEST CONTROL INTERFACE (TCI)

This section describes the Bluetooth Test Control Interface (TCI). The TCI provides a uniform method of accessing the upper interface of the implementation being tested. This facilitates the use of a standardized interface on test equipment used for formal Qualification of implementations.

2.1 INTRODUCTION

2.1.1 Terms used

Conformance testing	Testing according to the applicable procedures given in the Bluetooth Protocol Test Specifications and the Bluetooth Profile Conformance Test Specification when tested against a test system.
HCI	Host Controller Interface
IUT	Implementation Under Test: An implementation of one or more Bluetooth protocols and profiles which is to be studied by testing. This term is used when describing the test concept for products and components equipped with Bluetooth wireless technology as defined in the PRD.
PRD	Bluetooth Qualification Program Reference Document: This document is maintained by the Bluetooth Qualification Review Board and is the reference to specify the functions, organization and processes inside the Bluetooth Qualification program.
TCI	Test Control Interface: The interface and protocol used by the test equipment to send and receive messages to and from the upper interface of the IUT.

2.1.2 Usage of the interface

For all products and components equipped with Bluetooth wireless technology, conformance testing is used to verify the implemented functionality in the lower layers. Conformance testing of the lowest layers requires an upper tester to test the implementation sufficiently well.

In order to avoid that the tester will have to adapt to each and every product and component equipped with Bluetooth wireless technology, the use of the standardized TCI is mandated. This concept puts some burden upon the manufacturer of the IUT in terms of supplying an adapter providing the necessary conversion from/to the IUT's specific interface to the TCI. The adapter can consist of hardware, firmware and software. After qualification testing has been performed the TCI may be removed from the product or component equipped with Bluetooth wireless technology. It is the manufacturer's option to remove it from the qualified product or component equipped with Bluetooth wireless technology.



The TCI is used when qualifying the implemented functionality of the:

- Baseband layer, BB
- Link Manager layer, LM

If support of the Host Controller Interface is claimed by the manufacturer the TCI is used to qualify it.

2.2 TCI CONFIGURATIONS

This section describes the test configurations used when verifying the different Bluetooth requirements. Each layer in the Bluetooth stack is qualified using the procedures described in the layer specific test specification.

2.2.1 Bluetooth RF requirements

For qualification of the Bluetooth Radio Frequency requirements the defined Test Mode is used, see [Section 1 on page 231](#).

Similar to TCI, the specific test mode functionality may be removed from the product or component after qualification, at the discretion of the manufacturer.

2.2.1.1 Required interfaces

For RF qualification only the air interface is required, see [Figure 2.1](#). Depending on the physical design of the IUT it might be necessary to temporarily attach an RF connector for executing the RF tests. As stated in [Section 1 on page 231](#), the Test Mode shall be locally enabled on the IUT for security reasons. The implementation of this local enabling is not subject to standardization.

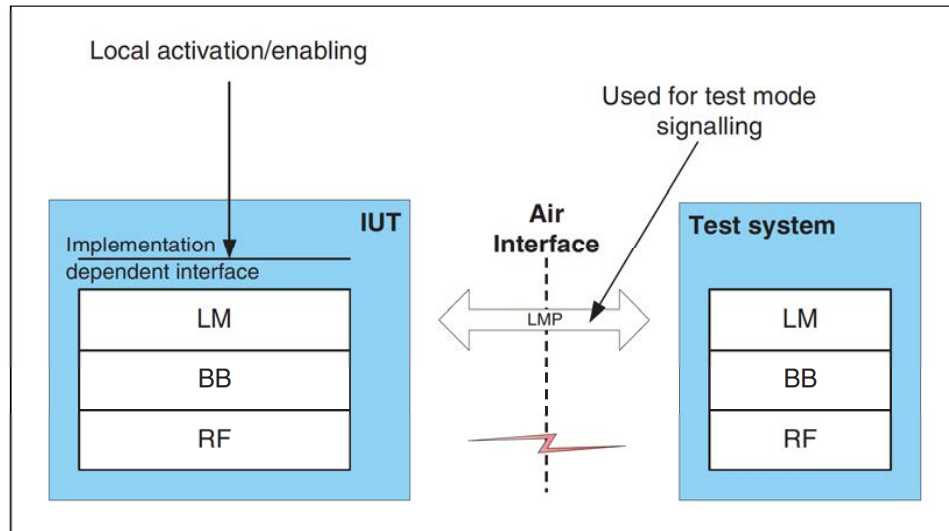


Figure 2.1: General test set-up for RF qualification

2.2.2 Bluetooth protocol requirements

Depending on which of the Bluetooth layers BB, LM or HCI are implemented in the product subject to qualification, the amount of testing needed to verify the Bluetooth protocol requirements differs. Also, the TCI used during the qualification is slightly different. The commands and events necessary for qualification are detailed in the test specifications and only those commands indicated in the test cases to be executed need be implemented.

For other protocols in the Bluetooth stack the TCI is not used. An implementation specific user interface is used to interact with the IUT's upper interface.

2.2.2.1 Required interfaces

For BB, LM and HCI qualification both the air interface of the IUT and the TCI are required. For other protocols both the air interface and the user interface are used as described in the test specification.



2.2.3 Bluetooth profile requirements

For each Bluetooth profile for which conformance is claimed, profile qualification testing is performed to verify the Bluetooth profile requirements. With higher layer protocols the TCI is not used during testing. A user interface specific to the implementation is used to interact with the IUT's upper interface.

2.2.3.1 Required interfaces

For this type of qualification both the air interface and the user interface of the IUT are used as described in the test specification.

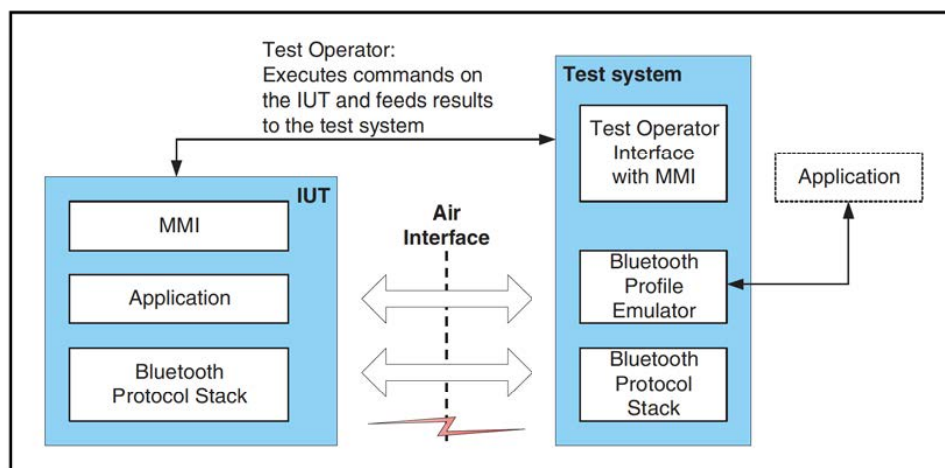


Figure 2.2: General test set-up for profile qualification



2.3 TCI CONFIGURATION AND USAGE

This interface is semantically and syntactically identical to the HCI interface described in “[Host Controller Interface Functional Specification](#)” on [page 335](#)[vol. 3]. The complete HCI interface is not required in the TCI, but the subset of HCI commands and events necessary for verifying the functionality of the IUT. The exact set of commands and events is specified in the test specifications for the layers subject to testing.

It is worth emphasizing again the TCI is an adapter, logically attached to the upper interface of the IUT. As such the TCI adapts the standardized signalling described here to the implementation specific interface of the IUT.

2.3.1 Transport layers

The method used to convey commands and events between the tester and the IUT's upper interface can be either physical bearer or “software bearer”.

2.3.1.1 Physical bearer

It is recommended to use one of the transport layers specified for the HCI in. However, other physical bearers are not excluded and may be provided on test equipment. The use of a physical bearer is required for test cases active in category A. Please see the PRD for the definitions of test case categories. Please see the current active Test Case Reference List for the categorization of specific test cases.

2.3.1.2 Software bearer

There is no physical connection between the tester and the IUT's upper interface. In this case, the manufacturer of the IUT shall supply test software and hardware that can be operated by a test operator. The operator will receive instructions from the tester and will execute them on the IUT. The “software bearer” shall support the same functionality as if using the TCI with a physical bearer. Use of the “software bearer” shall be agreed upon between the manufacturer of the IUT and the test facility that performs the qualification tests. The test facilities can themselves specify requirements placed on such a “software bearer”. Furthermore, the use of a “software bearer” is restricted to test cases active in one of the three lower categories B, C and D.

2.3.2 Baseband and link manager qualification

For the qualification of the link control part of the Baseband layer and for the Link Manager layer, the TCI is used as the interface between the test system and the upper interface of the IUT. The test system accesses the upper interface of the IUT by sending HCI commands and receiving HCI events from the IUT as described in the HCI specification. The required functionality on the TCI depends on the IUT's implemented functionality of the BB and LM layers, and therefore which test cases are executed.

A schematic example in [Figure 2.3](#) shows the test configuration for BB and LM qualification of Bluetooth products which do not support HCI, and use a physical bearer for the TCI. In this example the Test Control (TC) Software represents what the manufacturer has to supply with the IUT when using an external test facility for qualification. The function of the TC Software is to adapt the implementation dependent interface to the TCI.

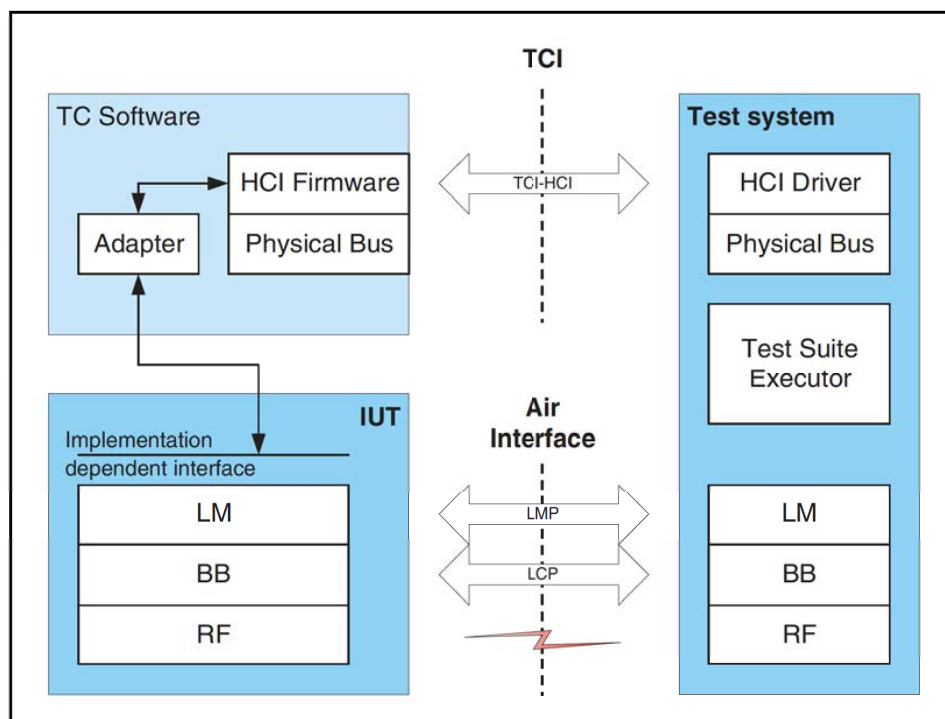


Figure 2.3: BB and LM qualification with TCI physical bearer



Figure 2.4 shows a schematic example of the test configuration for the same Bluetooth product using a “software bearer” for the TCI. Here the function of the Test Control Software is to represent the application that can be controlled by the test operator.

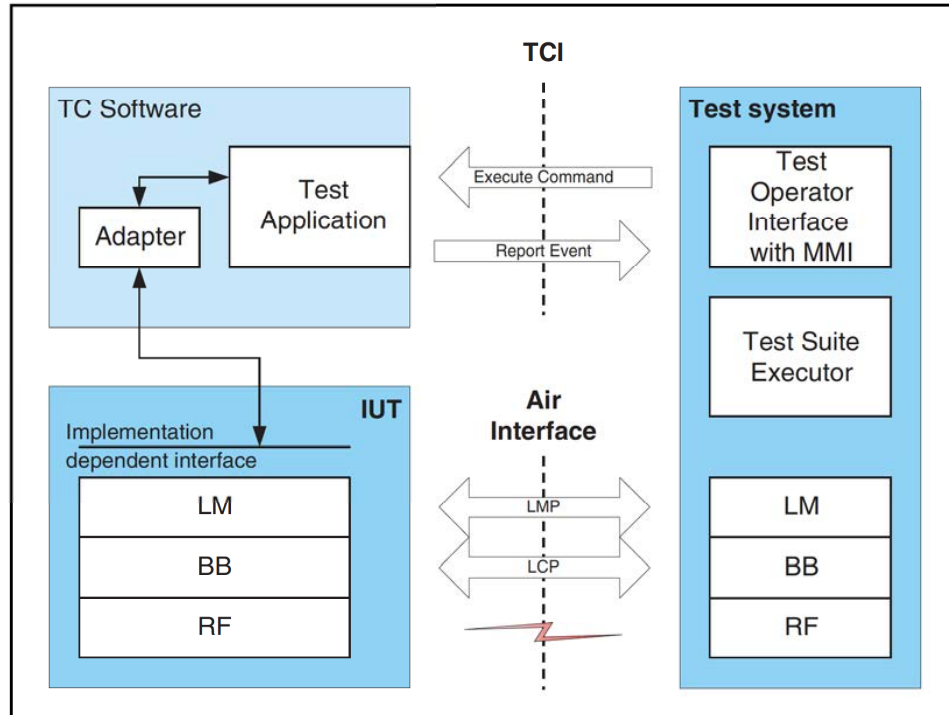


Figure 2.4: BB and LM qualification with “software bearer”



2.3.3 HCI qualification

The TCI may also be used for HCI signalling verification and qualification. The HCI signalling is only verified if support of the HCI functionality is claimed by the manufacturer.

A schematic example in [Figure 2.5](#) shows the test configuration for HCI qualification of Bluetooth products. As can be seen in the figure the implemented HCI is used as the interface to the tester.

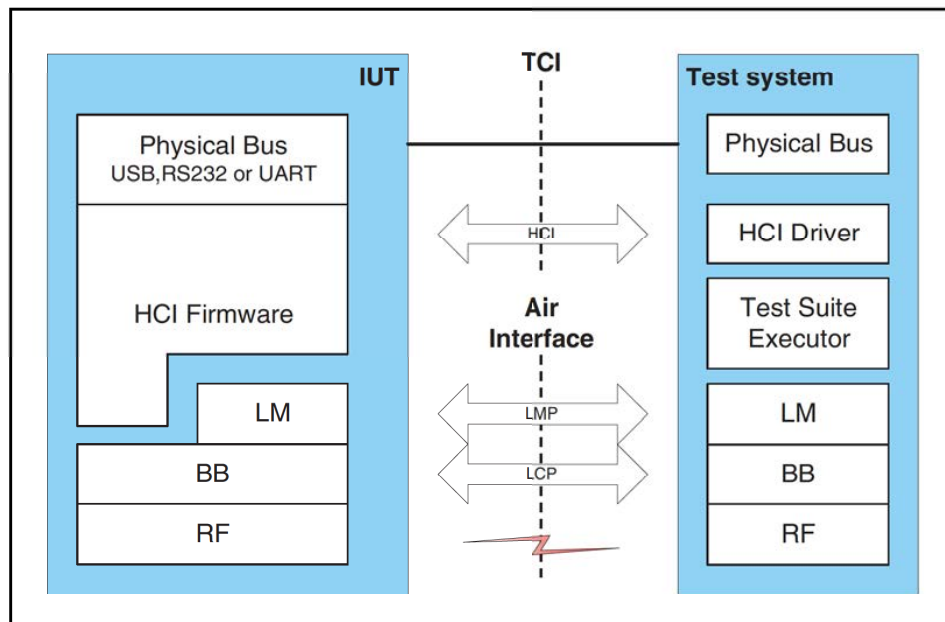


Figure 2.5: General test set-up for HCI qualification

