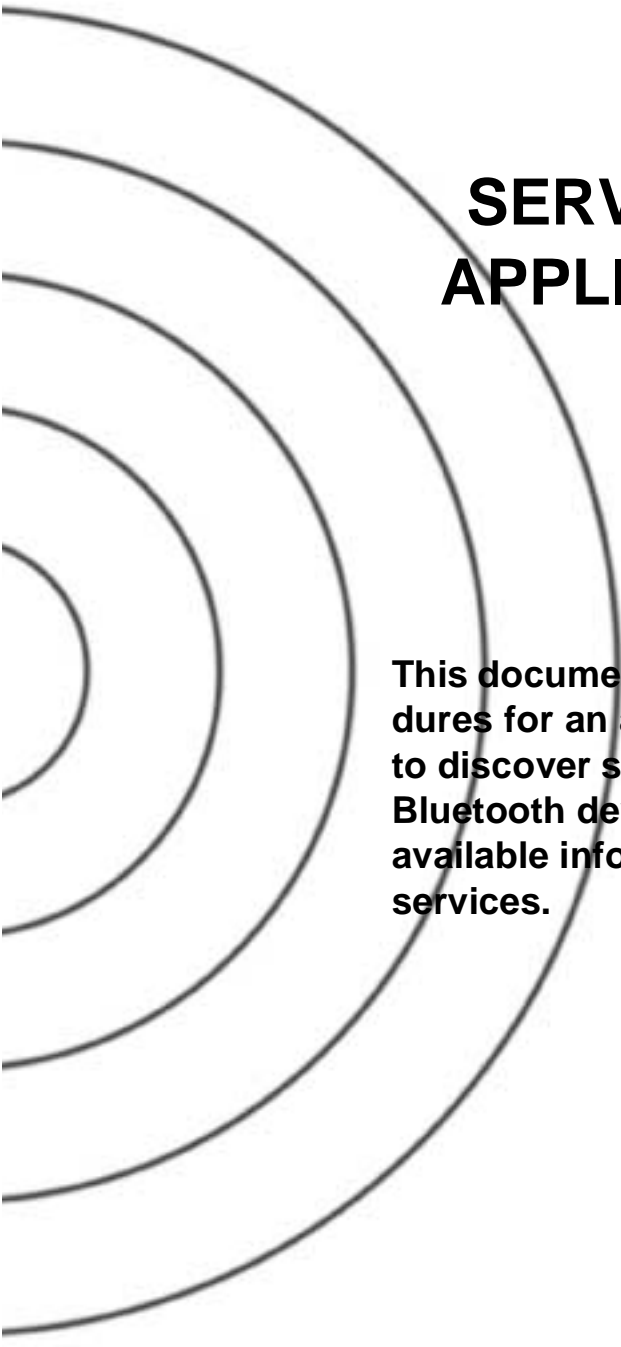


Part K:2

SERVICE DISCOVERY APPLICATION PROFILE



This document defines the features and procedures for an application in a Bluetooth device to discover services registered in other Bluetooth devices and retrieve any desired available information pertinent to these services.



CONTENTS

1	Introduction	66
1.1	Scope	66
1.2	Symbols and conventions	67
2	Profile overview	68
2.1	Profile stack	68
2.2	Configurations and roles	69
2.3	User requirements and scenarios	70
2.4	Profile fundamentals	71
2.5	Conformance	71
3	User interface aspects	72
3.1	Pairing	72
3.2	Mode selection	72
4	Application layer	73
4.1	The service discovery application	73
4.2	Service primitives abstractions.....	75
4.3	Message sequence charts (MSCs)	77
5	Service Discovery	79
5.1	An SDP PDU exchange example.....	80
6	L2CAP	82
6.1	Channel types	83
6.2	Signalling	83
6.3	Configuration options	83
6.3.1	Maximum Transmission Unit (MTU)	83
6.3.2	Flush Time-out	83
6.3.3	Quality of Service	84
6.4	SDP transactions and L2CAP connection lifetime	84
7	Link Manager	86
7.1	Capability overview	86
7.2	Error behavior	87
7.3	Link policy	87
8	Link control.....	88
8.1	Capability overview	88
8.2	Inquiry	89
8.3	Inquiry scan.....	90
8.4	Paging.....	90
8.5	Page scan	90
8.6	Error behavior	90



9	References.....	91
	9.1 Normative references	91
10	Definitions	92
11	Appendix A (Informative): Service primitives and the Bluetooth PDUs.....	93

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 an unambiguous 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

It is expected that the number of services that can be provided over Bluetooth links will increase in an undetermined (and possibly uncontrolled) manner. Therefore, procedures need to be established to aid a user of a Bluetooth-enabled device to sort the ever-increasing variety of services that will become available to him/her. While many of the Bluetooth-enabled services that may be encountered are currently unknown, a standardized procedure can still be put into place on how to locate and identify them.

The Bluetooth protocol stack contains a Service Discovery Protocol (SDP) [BT_SDP_spec:\[7\]](#) that is used to locate services that are available on or via devices in the vicinity of a Bluetooth enabled device. Having located what services are available in a device, a user may then select to use one or more of them. Selecting, accessing, and using a service is outside the scope of this document. Yet, even though SDP is not directly involved in accessing services, information retrieved via SDP facilitates service access by using it to properly condition the local Bluetooth stack to access the desired service.

The service discovery profile defines the protocols and procedures that shall be used by a service discovery application on a device to locate services in other Bluetooth-enabled devices using the Bluetooth Service Discovery Protocol (SDP). With regard to this profile, the service discovery application is a specific user-initiated application. In this aspect, this profile is in contrast to other profiles where service discovery interactions between two SDP entities in two Bluetooth-enabled devices result from the need to enable a particular transport service (e.g. RFCOMM, etc.), or a particular usage scenario (e.g. file transfer, cordless telephony, LAN AP, etc.) over these two devices. Service discovery interactions of the latter kind can be found within the appropriate Bluetooth usage scenario profile documents.

The service discovery in the other profile documents has a very narrow scope; e.g. learning about the protocols and related protocol parameters needed for accessing a particular service. Nevertheless, the fundamentals of the service discovery procedures covered in this profile document, and the use of the Bluetooth protocols in support of these procedures can be replicated in other profile documents as well. The only difference is that for the other profiles these procedures are initiated by application-level actions within the applications described by the corresponding profiles, as opposed to user-level actions for this profile.



SDP provides direct support for the following set of service inquiries:

- Search for services by service class;
- Search for services by service attributes; and
- Service browsing.

The generic service discovery application considered for this profile also covers the above service inquiry scenarios.

The former two cases represent searching for known and specific services. They provide answers to user questions like: “Is service A, or is service A with characteristics B and C, available?” The latter case represents a general service search and provides answers to questions like: “What services are available?” or “What services of type A are available?”

The above service inquiry scenarios can be realized two-fold:

- By performing the service searches on a particular device that a user ‘consciously’ has already connected to, and/or
- By performing the service searches by ‘unconsciously’ connecting to devices discovered in a device's vicinity.

Both of the above approaches require that devices need first to be discovered, then linked with, and then inquired about the services they support.

1.2 SYMBOLS AND CONVENTIONS

This profile uses the symbols and conventions specified in [Section 1.2](#) of the Generic Access Profile [\[3\]](#).

2 PROFILE OVERVIEW

2.1 PROFILE STACK

Figure 2.1 shows the Bluetooth protocols and supporting entities involved in this profile.

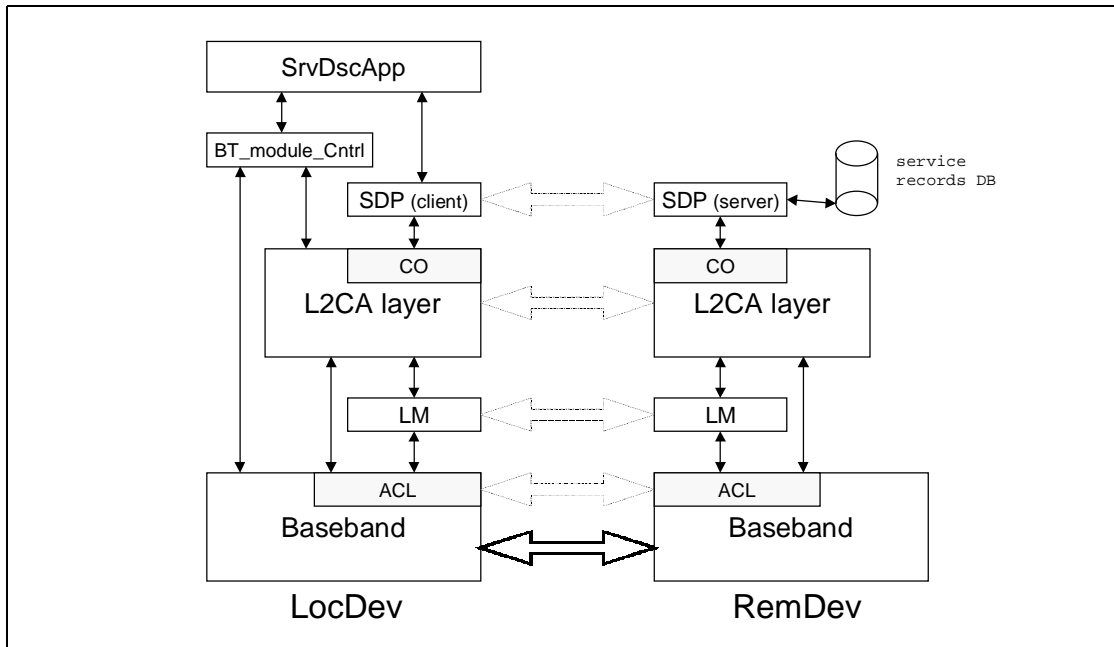


Figure 2.1: The Bluetooth protocol stack for the service discovery profile

The service discovery user application (SrvDscApp) in a local device (LocDev) interfaces with the Bluetooth SDP client to send service inquiries and receive service inquiry responses from the SDP servers of remote devices (RemDevs) BT_SDP_spec:[7]. SDP uses the connection-oriented (CO) transport service in L2CAP, which in turn uses the baseband asynchronous connectionless (ACL) links to ultimately carry the SDP PDUs over the air.

Service discovery is tightly related to discovering devices, and discovering devices is tightly related to performing inquiries and pages. Thus, the SrvDscApp interfaces with the baseband via the BT_module_Cntrl entity that instructs the Bluetooth module when to enter various search modes of operation.¹

1. The BT_module_Cntrl may be part of a Bluetooth stack implementation (and thus be shared by many Bluetooth-aware applications) or a 'lower part' of the SrvDscApp. Since, no assumptions about any particular stack or SrvDscApp implementations are made, the BT_module_Cntrl entity represents a logical entity separate from the SrvDscApp, which may or may not be part of the SrvDscApp itself, a stack component, or any other appropriate piece of code.



The service records database (DB) shown in [Figure 2.1](#) next to an SDP server is a logical entity that serves as a repository of service discovery-related information. The ‘physical form’ of this database is an implementation issue outside the scope of this profile.

2.2 CONFIGURATIONS AND ROLES

The following roles are defined in this profile:

- **Local device (LocDev):** A LocDev is the device that initiates the service discovery procedure. A LocDev must contain at least the *client* portion of the Bluetooth SDP architecture BT_SDP_spec:[7]. A LocDev contains the service discovery application (SrvDscApp) used by a user to initiate discoveries and display the results of these discoveries.
- **Remote Device(s) (RemDev(s)):** A RemDev is any device that participates in the service discovery process by responding to the service inquiries generated by a LocDev. A RemDev must contain at least the *server* portion of the Bluetooth SDP architecture BT_SDP_spec:[7]. A RemDev contains a service records database, which the server portion of SDP consults to create responses to service discovery requests.

The LocDev or RemDev role assigned to a device is neither permanent nor exclusive. A RemDev may also have a SrvDscApp installed into it as well as an SDP client, and a LocDev may also have an SDP server. In conjunction with which device has an SrvDscApp installed, an SDP-client installed, and an SDP-server installed, the assignment of devices to the above roles is relative to each individual SDP (and related) transaction and which device initiates the transaction. Thus, a device could be a LocDev for a particular SDP transaction, while at the very same time be a RemDev for another SDP transaction.

With respect to this profile, a device without a UI (directly or indirectly available) for entering user input and returning the results of service searches is not considered as a candidate for a LocDev. Nevertheless, even if such a device is not considered as a candidate for a LocDev, the procedures presented in the following sections can still apply if applications running in such a device need to execute a service discovery transaction.

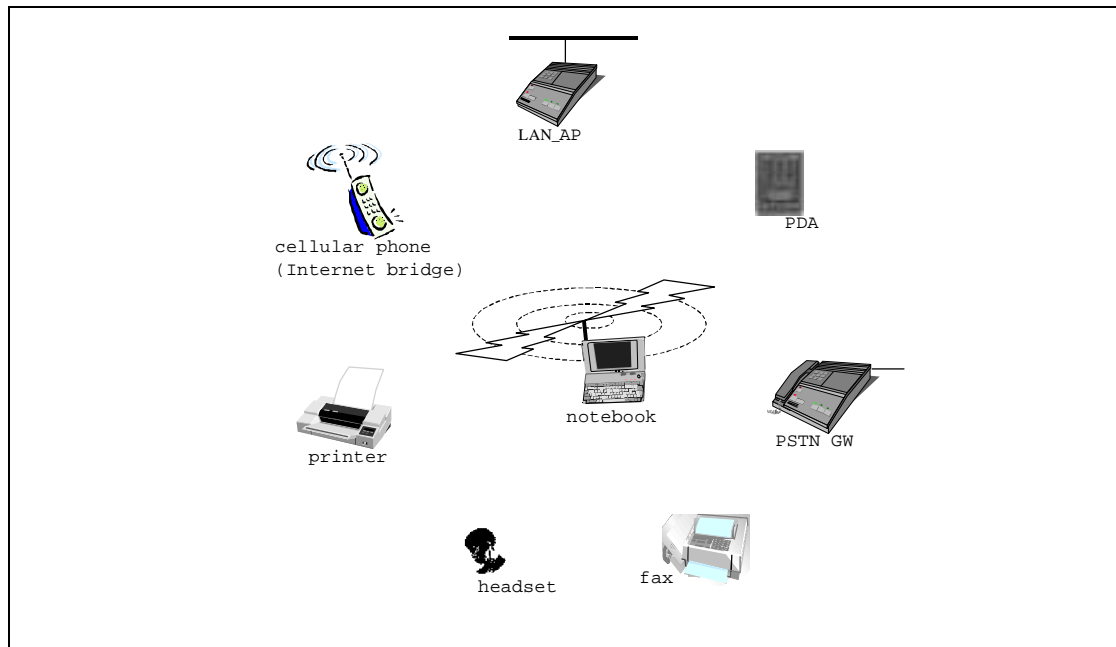


Figure 2.2: A typical service discovery scenario

The figure above shows a local device (the notebook) inquiring for services among a plethora of remote devices.

2.3 USER REQUIREMENTS AND SCENARIOS

The scenarios covered by this profile are the following:

- Search for services by service class,
- Search for services by service attributes, and
- Service browsing.

The first two cases represent searching for known and specific services, as part of the user question “Is service A, or is service A with characteristics B and C, available?” The latter case represents a general service search that is a response to the user question “What services are available?”

This profile implies the presence of a Bluetooth-aware, user-level application, the SrvDscApp, in a LocDev that interfaces with the SDP protocol for locating services. In this aspect, this profile is unique as compared to other profiles. It is a profile that describes an application that interfaces to a specific Bluetooth protocol to take full advantage of it for the direct benefit of an end-user.



2.4 PROFILE FUNDAMENTALS

Before any two Bluetooth-equipped devices can communicate with each other the following may be needed:

- The devices need to be powered-on and initialized. Initialization may require providing a PIN for the creation of a link key, for device authorization and data encryption.
- A Bluetooth link has to be created, which may require the discovery of the other device's BD_ADDR via an inquiry process, and the paging of the other device.

While it may seem natural to consider a LocDev serving as a Bluetooth master and the RemDev(s) serving as Bluetooth slave(s), there is no such requirement imposed on the devices participating in this profile. Service discovery as presented in this document can be initiated by either a master or a slave device at any point for which these devices are members of the same piconet. Also, a slave in a piconet may possibly initiate service discovery in a new piconet, provided that it notifies the master of the original piconet that it will be unavailable (possibly entering the hold operational mode) for a given amount of time.²

The profile does not require the use of authentication and/or encryption. If any of these procedures are used by any of the devices involved, service discovery will be performed only on the subset of devices that pass the authentication and encryption security 'roadblocks' that may impose to each other. In other words, any security restrictions for SDP transactions are dictated by the security restrictions already in place (if any) on the Bluetooth link.

2.5 CONFORMANCE

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 to 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.

2. Recall that a master of a piconet cannot initiate a new piconet. Since a piconet is ultimately identified by the BD_ADDR and the Bluetooth clock of its master, the latter piconet will be identical to and indistinguishable from the former.

3 USER INTERFACE ASPECTS

3.1 PAIRING

No particular requirements regarding pairing are imposed by this profile. Pairing may or may not be performed. Whenever a LocDev performs service discovery against as yet 'unconnected' RemDev(s), it shall be the responsibility of the SrvDscApp to allow pairing prior to connection, or to by-pass any devices that may require pairing first. This profile is focused on only performing service discovery whenever the LocDev can establish a legitimate and useful baseband link³ with RemDev(s).

3.2 MODE SELECTION

This profile assumes that, under the guidance of the SrvDscApp, the LocDev shall be able to enter the inquiry and/or page states. It is also assumed that a RemDev with services that it wants to make available to other devices (e.g. printer, a LAN DAP, a PSTN gateway, etc.) shall be able to enter the inquiry scan and/or page scan states. For more information about the inquiry and page related states see [Section 8](#).

Since the SrvDscApp may also perform service inquiries against already connected RemDevs, it is not mandatory according to the profile that a LocDev always be the master of a connection with a RemDev. Similarly, a RemDev may not always be the slave of a connection with a LocDev.

3. A legitimate and useful baseband link is a Bluetooth baseband link that is properly authenticated and encrypted (if so desired), whenever any of these options are activated by any of the devices participating in this profile.

4 APPLICATION LAYER

4.1 THE SERVICE DISCOVERY APPLICATION

In this subsection, the operational framework of the SrvDscApp is presented.⁴ Figure 4.1 shows alternative possibilities for a SrvDscApp.

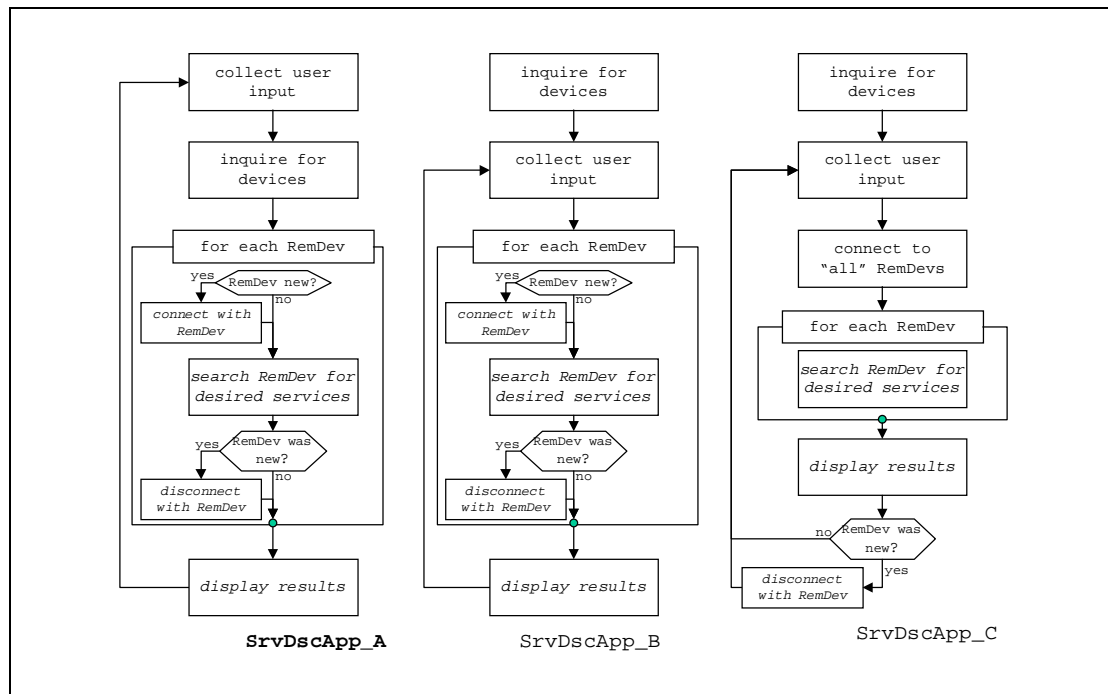


Figure 4.1: Three possible SrvDscApps

The SrvDscApp alternatives shown in Figure 4.1, which are not exhaustive by any means, achieve the same objectives but they follow different paths for achieving them. In the first alternative (SrvDscApp_A), the SrvDscApp on a LocDev inquires its user to provide information for the desired service search. Following this, the SrvDscApp searches for devices, via the Bluetooth inquiry procedure. For each device found, the LocDev will connect to it, perform any necessary link set-up, see related procedures in Generic Access Profile [3], and then inquire it for the desired services. In the second alternative (SrvDscApp_B), the inquiry of devices is done prior to collecting user input for the service search.⁵

4. This profile does not dictate any particular implementation for a SevDisApp. It only presents the procedures needed to achieve its objectives.

5. Device inquiries may even occur by means outside the scope of a particular SrvDscApp implementation. But, since such other means are not guaranteed to exist, it is recommended that the SrvDscApp activates device inquiries too.



In the first two alternatives, page, link creation, and service discovery are done sequentially on a per RemDev basis; i.e., the LocDev does not page any new RemDev prior to completing the service search with a previous RemDev and (if necessary) disconnecting from it. In the last alternative (SrvDscApp_C), the LocDev, under the control of the SrvDscApp, will first page all RemDevs, then will create links with all of these devices (up to a maximum of 7 at a time), and then inquire all the connected devices for the desired services.

Just as an example, we focus on a SrvDscApp similar to the one represented by the SrvDscApp_A in [Figure 4.1](#). In summary, SrvDscApp (for ease of notation, the suffix '_A' has been dropped) has the following features:

- The SrvDscApp activates Bluetooth inquiries following a user request for a service search,
- For any new RemDev found following an inquiry, the SrvDscApp will finish service discovery and terminate its link against this device prior to attempting to connect to the next RemDev,
- For any RemDev already connected, the LocDev does not disconnect following service discovery, and
- The user of the SrvDscApp has the option of a trusted and untrusted mode of operation, whereby the SrvDscApp permits connections –
 - a) only with trusted RemDev, or
 - b) with any of the devices above plus any newly discovered RemDevs that require nothing more beyond possibly pairing with the default all-zero PIN, or
 - c) with any of the devices above, plus any additional RemDev for which the user explicitly enters a non-zero PIN.

The above options have to do with the degree of user involvement in configuring and interacting with the SrvDscApp and setting the security levels that the user is willing to accept for the service searches. When selecting options (a) or (b), then for the devices with which no legitimate connections can be established, it is assumed that the SrvDscApp ignores them without any cue to its user (however, this too is an implementation issue).

When a LocDev performs a service discovery search, it does so against three different types of RemDevs:

1. *trusted devices*: These are devices that are currently not connected with the LocDev but the LocDev device has already an established trusted relation with.
2. *unknown (new) devices*: These are untrusted devices that are currently not connected with the LocDev.
3. *connected devices*: These are devices that are already connected to the LocDev.



To discover type 1 or 2 RemDevs, the SrvDscApp needs to activate the Bluetooth inquiry and/or page processes. For type 3 RemDevs, the latter processes are needed. To perform its task, SrvDscApp needs to have access to the BD_ADDR of the devices in the vicinity of a LocDev, no matter whether these devices have been located via a Bluetooth inquiry process or are already connected to the LocDev. Thus, BT_module_Cntr in a LocDev shall maintain the list of devices in the vicinity of the LocDev and shall avail this list to the SrvDscApp.

4.2 SERVICE PRIMITIVES ABSTRACTIONS

This section briefly describes the functionality of a SrvDscApp. This functionality is presented in the form of service primitive abstractions that provide a formal framework for describing the user expectations from a SrvDscApp. It is assumed that the underlying Bluetooth stack can meet the objectives of these service primitive abstractions directly or indirectly.⁶ The exact syntax and semantics of the service primitive abstractions (or simply “service primitives”) may be platform-dependent (e.g. an operating system, a hardware platform, like a PDA, a notebook computer, a cellular phone, etc.) and are beyond the scope of this profile. However, the functionality of these primitives is expected to be available to the SrvDscApp to accomplish its task.

Table 4.1 contains a minimum set of enabling service primitives to support a SrvDscApp. Low-level primitives like **openSearch(.)** or **closeSearch(.)** are not shown and are assumed to be part of the implementation of the primitives shown whenever necessary. Different implementations of the Bluetooth stack shall (at a minimum) enable the functions that these service primitives provide. For example, the **serviceSearch(.)** service primitive permits multiple identical operations to be handled at once. A stack implementation that requires an application to accomplish this function by iterating through the multiple identical operations one-at-a-time will be considered as enabling the function of this service primitive.⁷ The service primitives shown next relate only to service primitives whose invocation result or relate to an over-the-air data exchange using the Bluetooth protocols. Additional service primitives can be envisioned relating to purely local operations like *service registration*, but these primitives are outside the scope of this profile.

6. These service primitive abstractions do *not* represent programming interfaces, even though they may be related to them. The word ‘directly’ is used to describe the possibility that the described function is the result of a single appropriate call of the underlying Bluetooth stack implementation. The word ‘indirectly’ is used to describe the possibility that the described function can be achieved by combining the results from multiple appropriate calls of the underlying Bluetooth stack implementation.

7. Even though the service primitives presented in this profile are assumed to act upon a local device for accessing *physically* remote devices, they are general enough to apply in cases where the ‘remote device’ characterization is only a logical concept; i.e. inquired service records and service providers are located within the same device that invokes these primitives. This general situation is outside the scope of this profile.



service primitive abstraction	resulted action
serviceBrowse (LIST(<i>RemDev</i>) LIST(<i>RemDevRelation</i>) LIST(<i>browseGroup</i>) <i>getRemDevName</i> <i>stopRule</i>)	a search for services (service browsing) that belong to the list of <i>browseGroup</i> services in the devices in the list of <i>RemDevs</i> ; the search may be further qualified with a list of <i>RemDevRelation</i> parameters, whereby a user specifies the trust and connection relation of the devices to be searched; e.g. search only the devices that are in the <i>RemDev</i> list for which there is a trust relation already established; when the <i>getRemDevName</i> parameter is set to “yes,” the names of the devices supporting the requested services are also returned; the search continues until the stopping rule <i>stopRule</i> is satisfied
serviceSearch (LIST(<i>RemDev</i>) LIST(<i>RemDevRelation</i>) LIST(<i>searchPattern</i> , <i>attributeList</i>) <i>getRemDevName</i> <i>stopRule</i>)	a search whether the devices listed in the list of <i>RemDevs</i> support services in the requested list of services; each service in the list must have a service search pattern that is a superset of the <i>searchPattern</i> ; for each such service the values of the attributes contained in the corresponding <i>attributeList</i> are also retrieved; the search may be further qualified with a list of <i>RemDevRelation</i> parameters, whereby a user specifies the trust and connection relation of the devices to be searched (e.g. search only the devices that are in the <i>RemDev</i> list for which there is a trust relation already established); when the <i>getRemDevName</i> parameter is set to “yes,” the names of the devices supporting the requested services are also returned; the search continues until the stopping rule <i>stopRule</i> is satisfied
enumerateRemDev (LIST(<i>classOfDevice</i>) <i>stopRule</i>)	a search for <i>RemDev</i> in the vicinity of a <i>LocDev</i> ; <i>RemDev</i> searches may optionally be filtered using the list of <i>classOfDevice</i> (e.g. LAN APs); the search continues until the stopping rule <i>stopRule</i> is satisfied
terminatePrimitive (<i>primitiveHandle</i> <i>returnResults</i>)	a termination the actions executed as a result of invoking the services primitive identified by the <i>primitiveHandle</i> ; [*] optionally, this service primitive may return any partially accumulated results related to the terminated service primitive

Table 4.1: Service primitives in support of *SrvDscApp*

*. It is assumed that each invocation of a service primitive can be identified by a *primitiveHandle*, the realization of which is implementation-dependent.

The *stopRule* parameter is used to guarantee a graceful termination of a service search. It could represent the number of search items found, or the duration of search, or both. A Bluetooth stack implementation may not expose this parameter, in which case it should provide guarantees that all searches terminate within a reasonable amount of time, for example, say, 120sec.



The **enumerateRemDev(.)** service primitive is directly related to the inquiry mode of operation for the baseband. It also relates to the collection of RemDev that a LocDev is currently connected with. This service is exported to the SrvDscApp via the BT_module_Cntr, see [Figure 2.1](#). The interface between BT_module_Cntr and baseband is for activating Bluetooth inquiries and collecting the results of these inquiries. The interface between the BT_module_Cntrl and (an) L2CAP (implementation) is for keeping track of the RemDev that currently are connected to the LocDev.

The result of the **enumerateRemDev(.)** service primitive can be used with the **serviceSearch(.)** to search for desired services in the devices found. Once again, based on the implementation of the Bluetooth stack, this service primitive may not be provided explicitly, but its service may be provided within other service primitives; e.g. the **serviceSearch(.)**.

Missing primitive parameters shall be interpreted (whenever appropriate) as a general service search on the remaining parameters. For example, if the LIST(*RemDev*) parameter is missing from the **serviceSearch(.)**, it means that the search shall be performed against any device found in the vicinity of a LocDev. In this case, the first two service primitives may be combined to a single one.

The above service primitives return the requested information, whenever found. Based on the way that these service primitives are supported by a Bluetooth stack implementation, the results of a search may directly return by the corresponding calling function, or a pointer to a data structure may be returned that contains all the relevant information. Alternatively, a Bluetooth stack implementation may have altogether different means for providing the results of a search.

4.3 MESSAGE SEQUENCE CHARTS (MSCS)

This profile is concerned with three distinct Bluetooth procedures. Device discovery, device name discovery, service discovery. Note that each one of these procedures does not preclude any other; e.g. to connect to a RemDev, a LocDev may have to first discover it, and it may also ask for its name. The MSCs relating to the first two procedures (i.e., device and name discovery) are provided in section 2 of LM/HCI_MSCs:[\[6\]](#). Sections 3, 4.1 and 4.2 of LM/HCI_MSCs:[\[6\]](#) provide the MSCs relating to the third procedure (i.e., service discovery). See also section 4 of BT_LM_spec:[\[4\]](#). The first two procedures do not require host intervention, while the third does.

[Figure 4.2](#) summarizes the key message exchange ‘phases’ encountered during the execution of this profile. Not all procedures are present at all times, and not all devices need to go through these procedures all the time. For example, if authentication is not required, the authentication phase in the figure will not be executed. If the SrvDsvApp needs to inquire for services on a specific RemDev with which the LocDev is currently connected, inquiries and pages

may not be executed. In the figure, the conditions under which particular phases are executed or not are also provided.

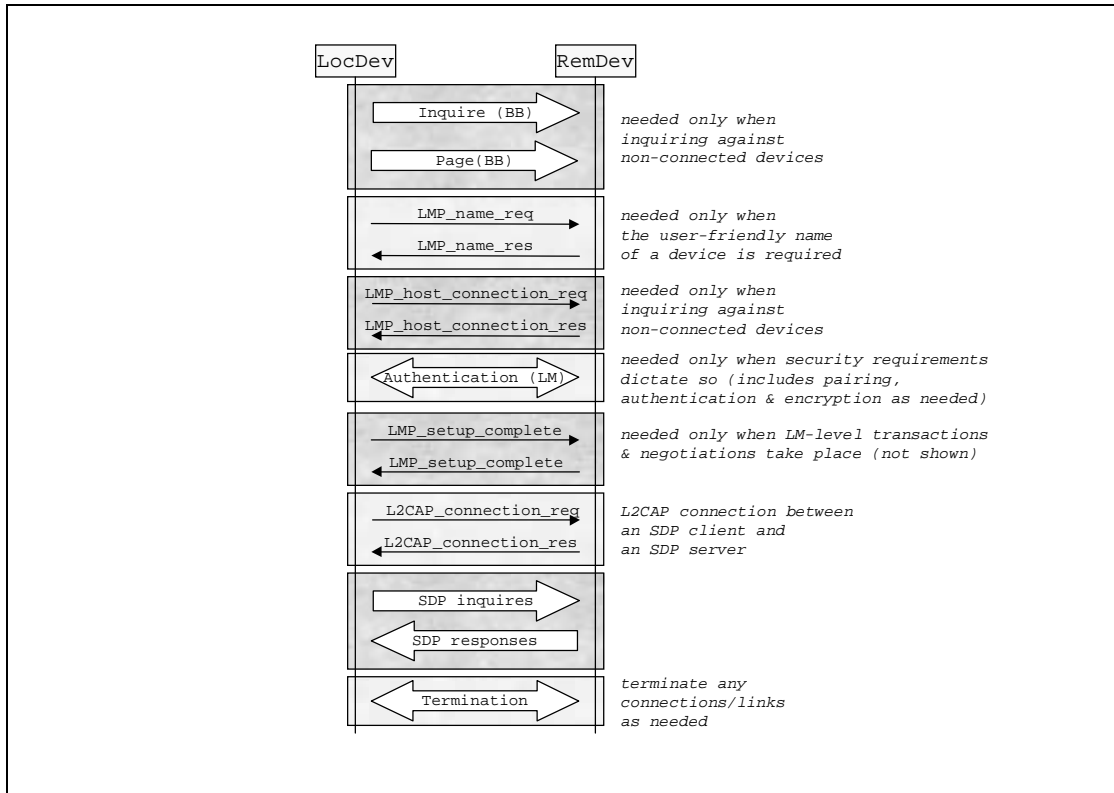


Figure 4.2: Bluetooth processes in support of this profile

In addition to the MSC in [Figure 4.2](#), Annex A shows what Bluetooth procedures and PDUs are needed to support the service primitives presented in [Section 4.2](#).



5 SERVICE DISCOVERY

The service discovery application does not make use of SDP as a means of accessing a service, but rather as a means of informing the user of a LocDev about the services that are available to his/her device by (and possibly via) RemDev(s). BT-aware applications running in a local device can also use the procedures described in this and the following sections to retrieve any pertinent information that will facilitate the application in accessing a desired service in a remote device.

Table 5.1 shows the SDP feature requirements in a LocDev and in a RemDev.

	SDP feature	Support in LocDev	Support in RemDev
1.	SDP client	M	O
2.	SDP server	O	M

Table 5.1: SDP feature requirements

Table 5.2 shows the SDP PDUs can be exchanged between devices following this profile.

SDP PDUs	Ability to Send		Ability to Receive	
	LocDev	RemDev	LocDev	RemDev
SDP_ErrorResponse	C1	M	M	C1
SDP_ServiceSearchRequest	M	C1	C1	M
SDP_ServiceSearchResponse	C1	M	M	C1
SDP_ServiceAttributeRequest	M	C1	C1	M
SDP_ServiceAttributeResponse	C1	M	M	C1
SDP_ServiceSearchAttributeRequest	M	C1	C1	M
SDP_ServiceSearchAttributeResponse	C1	M	M	C1
<i>Comments:</i>				
[C1]: With regard to this current profile, these PDU transmissions will not occur. Nevertheless, since a device could act as a LocDev on some occasions and as a RemDev on others, these PDU transmission may still take place between these devices.				

Table 5.2: Allowed SDP PDUs

5.1 AN SDP PDU EXCHANGE EXAMPLE

Figure 5.1 shows two examples of SDP PDU exchanges. In particular, it shows PDU exchange sequences for the inquiry and retrieval of any information pertinent to a particular Bluetooth profile.

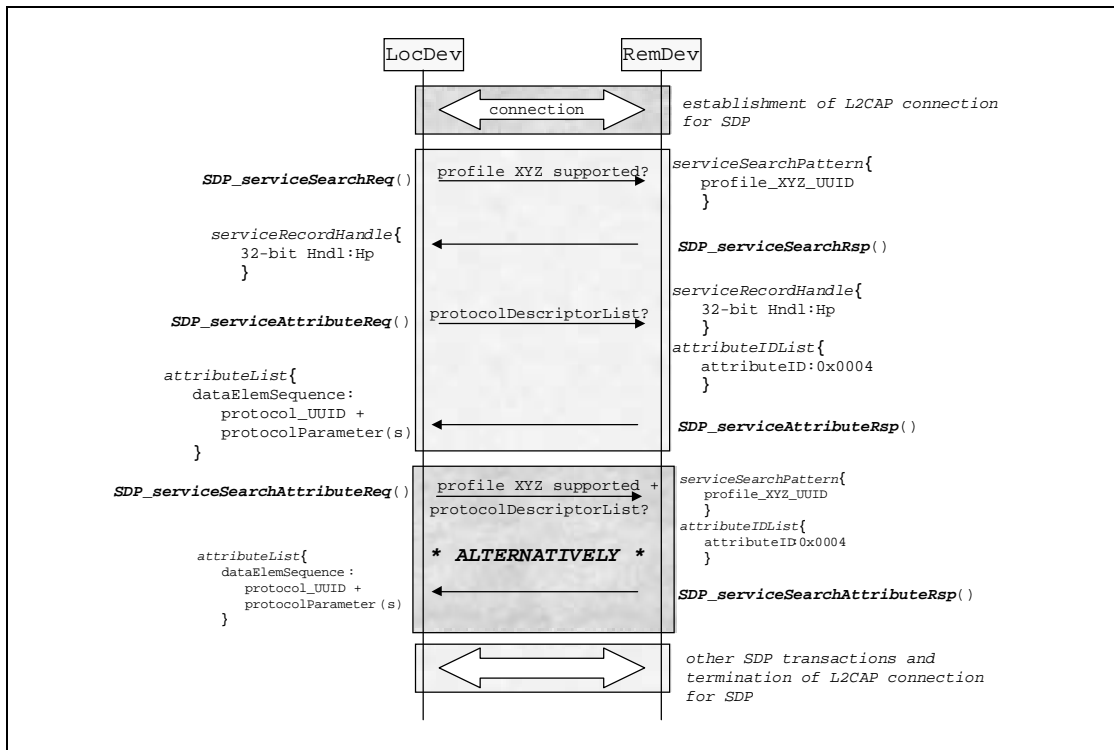


Figure 5.1: SDP PDU exchange examples for retrieving protocolDescriptorLists

For each PDU sent, the figure shows which device sends it (shown on the starting side of an arrow) and any relative information that this PDU carries (shown on the ending side of an arrow). Note that the LocDev sends request PDUs, while the RemDev sends back response PDUs.

Two alternatives are shown utilizing different SDP PDUs to ultimately retrieve the same information – the *protocolDescriptorList* attribute from devices that support a specific Bluetooth profile. With the first alternative, the desired information is derived in two steps.

- The LocDev sends an *SDP_serviceSearchReq* PDU which contains a service search pattern composed of the UUID associated with the desired profile; see section 4.3 of BT_ASN:[2]. The desired profile (profile ‘XYZ’) is identified by its UUID, denoted in the figure as ‘profile_XYZ_UUID.’ In its response PDU, the SDP server returns one or more 32-bit service record handles whose corresponding service records contain the ‘profile_XYZ_UUID’ UUID. In the figure, only one such handle is shown, denoted as ‘prHndl’.
- The LocDev then enters prHndl in an *SDP_serviceAttribute* PDU together with one or more attribute IDs. In this example, the attribute of interest is the



protocolDescriptorList, whose attribute ID is 0x0004. The SDP server then, in its response, returns the requested protocol list.

In the event that no service record containing the desired service search pattern is found in the SDP server, the *SDP_serviceSearchResp* PDU will contain an empty *serviceRecordHandleList* and a *totalServiceRecordCount* parameter set to its minimum value; see section 4.5.2 of BT_SDP_spec:[7].

If the desired attributes do not exist in the SDP server, the *SDP_serviceAttributeResp* PDU will contain an empty *attributeList* and an *attributeListByteCount* parameter set to its minimum value, see section 4.6.2 of BT_SDP_spec:[7].

With the second alternative, the desired attributes are retrieved in one step:

- The LocDev sends an *SDP_serviceSearchAttributeReq* PDU where both the desired profile is included (service search pattern: profile_XYZ_UUID) and the desired attribute(s) is provided (attribute ID: 0x0004). In its response the SDP server will provide the requested attribute(s) from the service record(s) that matches the service search pattern.

In case no service record containing the desired service search pattern and/or the desired attribute(s) is found in the SDP server, the *SDP_serviceSearchAttributeResp* PDU will contain an empty *attributeLists* and an *attributeListsByteCount* parameter set to its minimum value, see section 4.7.2 of BT_SDP_spec:[7].

While, in the example in [Figure 5.1](#), only very few service attributes are shown retrieved by the SDP client, additional information could and should be requested. Particularly in cases where service information is to be cached for future use, an SDP client should also request any pertinent information that can aid in assessing whether cached information has become stale. The service attributes *serviceDatabaseState*, *serviceRecordState*, and *serviceInfoTimeToLive* have been defined for this purpose in BT_SDP_spec:[7]; see sections 5.2.4, 5.1.3 and 5.1.8 respectively.

6 L2CAP

The following text, together with the associated subclauses, defines the mandatory requirements with regard to this profile.

	L2CAP procedure	Support in LocDev	Support in RemDev
1.	Channel types		
	Connection-oriented channel	M	M
	Connectionless channel	X1	X1
2.	Signalling		
	Connection Establishment	M	C1
	Configuration	M	M
	Connection Termination	M	C2
	Echo	M	M
	Command Rejection	M	M
3.	Configuration Parameter Options		
	Maximum Transmission Unit	M	M
	Flush Time-out	M	M
	Quality of Service	O	O
Comments:			
<p>[X1]: This feature is not used in this profile, but its use by other applications running simultaneously with this profile is not excluded.</p> <p>[C1]: An SDP server shall not (and cannot) initiate an L2CAP connection for SDP transactions. Nevertheless, the device that the SDP server resides in may also have an SDP client that may initiate an L2CAP connection for SDP transactions. Such action does not contradict the execution of this profile. In any case, a RemDev shall be able to process incoming requests for connection establishment.</p> <p>[C2] Under normal operation, an SDP server shall not initiate the process of terminating an L2CAP connection for SDP. However, exceptional cases, such as when a RemDev shuts down during the execution on an SDP transaction, cannot be excluded. In such a case, prior to the final power-off, the RemDev may gracefully (or not!) terminate all its active L2CAP connections by sending connection termination PDUs. In any case, a RemDev shall always be able to process incoming requests for connection termination.</p>			

Table 6.1: L2CAP procedures



6.1 CHANNEL TYPES

In this profile, only connection-oriented channels shall be used. In particular, no L2CAP broadcasts are to be used for this profile.

6.2 SIGNALLING

For the purpose of retrieving SDP-related information, only a LocDev can initiate an L2CAP connection request and issue an L2CAP connection request PDU; for exceptions, see comments C1 and C2 on [Table 6.1](#). Likewise with the corresponding L2CAP connection terminations, and the same exceptional comments C1 and C2 on [Table 6.1](#) apply. Other than that, SDAP does not impose any additional restrictions or requirements on L2CAP signalling.

In the PSM field of the Connection Request packet, the value 0x0001 (see section 5.2 of BT_L2CAP_spec:[\[5\]](#)) shall be used to indicate the request for creation of an L2CAP connection for accessing the SDP layer.

6.3 CONFIGURATION OPTIONS

This section describes the usage of configuration options in the service discovery profile.

6.3.1 Maximum Transmission Unit (MTU)

This profile does not impose any additional restrictions to MTU beyond the ones stated in section 6.1 of BT_L2CAP_spec:[\[5\]](#). If no MTU negotiation takes place, the default MTU value in section 6.1 of BT_L2CAP_spec:[\[5\]](#) shall be used.

For efficient use of the communication resources, the MTU shall be selected as large as possible, while respecting any physical constraints imposed by the devices involved, and the need that these devices continue honoring any already agreed upon QoS contracts with other devices and/or applications. It is expected that during the lifetime of an L2CAP connection for SDP transactions (also referred to as the 'SDP session', see [Section 6.4](#)) between two devices, any one of these devices may become engaged in an L2CAP connection with another device and/or application. If this new connection has 'non-default' QoS requirements, the MTU for the aforementioned SDP session is allowed to be re-negotiated during the lifetime of this SDP session, to accommodate the QoS constraints of the new L2CAP connection.

6.3.2 Flush Time-out

The SDP transactions are carried over an L2CAP reliable channel. The flush time-out value (see section 6.2 of BT_L2CAP_spec:[\[5\]](#)) shall be set to its default value 0xFFFF.

6.3.3 Quality of Service

The use of Quality of Service (QoS) and QoS negotiation is optional. If QoS is to be negotiated, the default settings in section 6.4 of BT_L2CAP_spec:[5] shall be used. In particular, SDP traffic shall be treated as a best-effort service type traffic.

6.4 SDP TRANSACTIONS AND L2CAP CONNECTION LIFETIME

While, in general, SDP transactions comprise a sequence of service request-and-response PDU exchanges, SDP itself constitutes a connectionless datagram service in that no SDP-level connections are formed prior to any SDP PDU exchange. SDP delegates the creation of connections on its behalf to the L2CAP layer. It is thus the responsibility of SDP – or, more correctly, of the SDP layer – to request the L2CAP layer to ‘tear down’ these connections on its behalf as well.

Since SDP servers are considered stateless, ‘tearing down’ an L2CAP connection after a service request PDU is sent (as a true connectionless service may imply) will be detrimental to the SDP transaction. Moreover, significant performance penalty will have to be paid if, for each SDP PDU transmission, a new L2CAP connection is to be created. Thus, L2CAP connections for SDP transactions shall last more than the transmission of a single SDP PDU.

An SDP *session* between an SDP client and an SDP server represents the time interval that the client and the server have the same L2CAP connection continuously present. A *minimal* SDP transaction will represent a single exchange of an SDP request PDU transmission from an SDP client to an SDP server, and the transmission of a corresponding SDP response PDU from the SDP server back to the SDP client. With respect to this profile, under normal operational conditions, the minimum duration of an SDP session shall be the duration of a minimal SDP transaction.

An SDP session may last less than the minimum required in the event of unrecoverable (processing or link) errors in layers below SDP in the LocDev and RemDev, or in the SDP layer and the service records database in the RemDev. An SDP session may also be interrupted by user intervention that may terminate the SDP session prior to the completion of an SDP transaction.

The above minimum duration of an SDP session guarantees smooth execution of the SDP transactions. For improved performance, implementers may allow SDP sessions to last longer than the minimum duration of an SDP session. As a general implementation guideline, an SDP session shall be maintained for as long as there is a need to interact with a specific device. Since the latter time is in general unpredictable, SDP implementations may maintain timers used to time periods of SDP transaction inactivity over a specific SDP session.



SDP implementations may also rely on explicit input received from a higher layer (probably initiated from the SrvDscApp itself) to open and close an SDP session with a particular device using low level primitives; e.g. **openSearch(.)** and **closeSearch(.)**. Finally, an implementation may permit users to interrupt an SDP session at any time, see the **terminatePrimitive(.)** service primitive in [Section 4.2](#).

Normally, an SDP session shall not terminate by a RemDev. Yet, such an event can indeed occur, either having the RemDev gracefully terminating the SDP session, using the L2CAP connection termination PDU, or abnormally terminating the SDP by stopping responding to SDP requests or L2CAP signalling commands. Such an event may be an indication of an exceptional condition that SDP client/server implementers should consider addressing for the smooth execution of this profile. If a termination event initiates from a RemDev, an SDP client may want to consider clearing any information obtained by this RemDev. Such an exceptional event may imply that the SDP server has (or is about to) shut-down, in which case any service information retrieved from this server should automatically become stale.



7 LINK MANAGER

7.1 CAPABILITY OVERVIEW

In this section, the LMP layer is discussed. In the table below, all LMP features are listed. The table shows which LMP features are mandatory to support with respect to this service discovery profile, which are optional and which are excluded. The reason for excluding features is that they may degrade operation of devices in this use case. Therefore, these features shall never be activated by a unit active in this use case.

If any of the rules stated below are violated, the units shall behave as defined in [Section 7.2](#).

Traffic generated during service discovery interactions has no particular QoS requirements. As such, no particular provision of the Bluetooth link is required to support this profile.

	LM Procedure	Support in LMP	Support in LocDev	Support in RemDev
1.	Authentication	M	C1	C1
2.	Pairing	M		
3.	Change link key	M		
4.	Change the current link key	M		
4.	Encryption	O	C1	C1
5.	Clock offset request	M		
6.	Timing accuracy information request	O		
7.	LMP version	M		
8.	Supported features	M		
9.	Switch of master slave role	O		
10.	Name request	M		
11.	Detach	M		
12.	Hold mode	O		
13.	Sniff mode	O		
14.	Park mode	O		
15.	Power control	O		

Table 7.1: LMP procedures



	LM Procedure	Support in LMP	Support in LocDev	Support in RemDev
16.	Channel quality driven DM/DH	O		
17.	Quality of service	M		
18.	SCO links	O	X1	X1
19.	Control of multi-slot packets	M		
20.	Concluding parameter negotiation	M		
21.	Host connection	M		
<p>Comments:</p> <p>[C1] No authentication or encryption is required specifically by this profile. This profile will, however, not attempt to change the existing operational settings for these procedures. Nevertheless, when this profile is executed all by itself, the default operational settings are: - authentication: no active - encryption: no active In the latter case, a LocDev will always comply with the security requirements imposed by a RemDev. If it cannot comply, it will bypass the RemDev.</p> <p>[X1]: This feature is not used in this profile, but its use by other applications running simultaneously with this profile is not excluded.</p>				

Table 7.1: LMP procedures

7.2 ERROR BEHAVIOR

If a unit tries to use a mandatory feature, and the other unit replies that it is not supported, the initiating unit shall send an LMP_detach PDU with detach reason "unsupported LMP feature."

A unit shall always be able to handle the rejection of the request for an optional feature.

7.3 LINK POLICY

There are no fixed master-slave roles for the execution of this profile.

This profile does not state any requirements on which low-power modes to use, or when to use them. It is up to the Link Manager of each device to decide and request special link features as seen appropriate.

8 LINK CONTROL

8.1 CAPABILITY OVERVIEW

The following table lists all features on the LC level

	Procedure	Support in baseband	Support in LocDev	Support in RemDev
1.	Inquiry	M	C1	
2.	Inquiry scan	M		C2
3.	Paging	M	C1	
4.	Page scan			
A	Type R0	M		C3
B	Type R1	M		C3
C	Type R2	M		C3
5.	Packet types			
A	ID packet	M		
B	NULL packet	M		
C	POLL packet	M		
D	FHS packet	M		
E	DM1 packet	M		
F	DH1 packet	M		
G	DM3 packet	O		
H	DH3 packet	O		
I	DM5 packet	O		
J	DH5 packet	O		
K	AUX packet	M	X1	X1
L	HV1 packet	M	X1	X1
M	HV2 packet	O	X1	X1
N	HV3 packet	O	X1	X1
O	DV packet	M	X1	X1
6.	Inter-piconet capabilities	O		
7.	Voice codec			

Table 8.1: LC features



	Procedure	Support in baseband	Support in LocDev	Support in RemDev
A	A-law	O	X1	X1
B	μ-law	O	X1	X1
C	CVSD	O	X1	X1
Comments:				
[C1]: This mandatory LC feature will be activated under the control of the SrvDscApp.				
[C2]: This mandatory LC feature is a settable device policy (outside the scope of this profile) that is activated whenever a device is to operate in a discoverable (public) mode.				
[C3] This mandatory LC feature is a settable device policy (outside the scope of this profile) that is activated whenever a device is to operate in a discoverable or connectable (private) mode.				
[X1]: These features are not used in this profile, but their use by other applications running simultaneously with this profile is not excluded.				

Table 8.1: LC features

For the next four subsections, it is assumed that a LocDev is to perform service searches with originally unconnected RemDevs. It thus needs to inquire for and page (or only page) these RemDevs. None of the following four subsections apply whenever a LocDev performs service searches with RemDevs to which it is already connected.

8.2 INQUIRY

Whenever instructed by the SrvDscApp, the LocDev shall advise its baseband to enter the inquiry state. Entry into this state may or may not be immediate, however, depending on QoS requirements of any already existing and ongoing connections.

The user of the SrvDscApp shall be able to set the criteria for the duration of an inquiry, see *stopRule* service primitive parameter in [Section 4.2](#). Nevertheless, the actual residence time in the inquiry state must comply with the recommendation given in section 10.7.3 of Bluetooth Baseband Specification [1].

When inquiry is invoked in a LocDev, the general inquiry procedure shall be used using a GIAC as described in [Section 6.1](#) of Bluetooth GAP_profile:[3].

Instead of a GIAC, an appropriate DIAC can be used to narrow down the scope of the inquiry. Since the only defined DIAC (referred to as the LIAC) does not reflect any specific device or service categories, the use of DIACs is of limited (but non-zero) benefit in this profile. In particular, the profile does not exclude (but neither does it encourage) performing inquiries according to the limited inquiry procedure described in [Section 6.2](#) of GAP_profile:[3]. The information contained in the Class of Device field in the FHS packet returned by the ‘inquired devices’ can be used as a filter to limit the number of devices to page and connect to for subsequent SDP transactions.



8.3 INQUIRY SCAN

Inquiry scans are device-dependent policies outside the scope of this profile. Devices that operate in a discoverable mode of operation, see [Section 4.1](#) of GAP_profile:[3], could be discovered by inquiries sent by other devices.

To be discovered by an inquiry resulting from a SrvDscApp action, a RemDev must enter inquiry scans using the GIAC; see general discoverable mode in [Section 4.1.3](#) of GAP_profile:[3]. A DIAC can be used instead of a GIAC. As previously mentioned, the use of DIACs are of limited (but non-zero) benefit in this profile. In particular, performing inquiry scans according to the limited discoverable procedure described in [Section 6.2](#) of GAP_profile:[3] is not excluded, but is not encouraged either.

8.4 PAGING

Whenever the SrvDscApp needs to connect to a specific RemDev for inquiring about its service records, the LocDev will advise its baseband to enter the page state. Entry into this state may or may not be immediate, however, depending on QoS requirements of any already existing and ongoing connections.

Depending on the paging class (R0, R1, or R2) indicated by a RemDev device, the LocDev shall page accordingly. The total residence time in the page state must comply with the recommendation given in section 10.6.3 of BT_BB_spec:[1]. For the pages, the 48-bit BD_ADDR of the RemDev must be used.

8.5 PAGE SCAN

Just like inquiry scans, page scans are device-dependent policies outside the scope of this profile. Devices that operate in a connectable mode of operation, see [Section 4.2.2](#) of GAP_profile:[3], could establish Bluetooth links with other devices from pages sent by these other devices. To establish a link with a RemDev, a LocDev must send a page that results from a SrvDscApp action using the RemDev's 48-bit BD_ADDR.

8.6 ERROR BEHAVIOR

Since most features on the LC level have to be activated by LMP procedures, errors will usually be caught at that layer. However, there are some LC procedures that are independent of the LMP layer, such as inquiry or paging. Misuse of such features is difficult or sometimes impossible to detect. There is no mechanism defined to detect or prevent such improper use.



9 REFERENCES

9.1 NORMATIVE REFERENCES

- [1] Baseband specification (see Volume 1, Part B)
- [2] Bluetooth Assigned Numbers
<http://www.bluetooth.org/assigned-numbers.htm>
- [3] Generic Access Profile (see Volume 2, Part K1)
- [4] Link Manager Protocol (see Volume 1, Part C)
- [5] Logical Link Control and Adaptation Protocol Specification (see Volume 1, Part D)
- [6] Message Sequence Charts between Host–Host Controller/Link Manager (see Volume 1, Appendix IX)
- [7] Service Discovery Protocol (see Volume 1, Part E)

10 DEFINITIONS

Term	Definition
conscious	(usually referred to) a process that requires the explicit intervention of a user to be accomplished
known	(with respect to a specific device) opposite to <i>unknown</i> ; a known device is not necessarily a <i>paired</i> device
new (RemDev)	(with regard to this profile) an additional remote device (RemDev) that is discovered during a Bluetooth inquiry, and that is not already connected to local device (LocDev)
private	a mode of operation whereby a device can only be found via Bluetooth baseband pages; i.e. it only enters page scans
public	a mode of operation whereby a device can be found via Bluetooth baseband inquiries; i.e. it enters into inquiry scans. A public device also enters into page scans (contrast this with <i>private</i>)
unconscious	opposite to <i>conscious</i>
unknown	(with respect to a specific device) any other device that a specific device has no record of



11 APPENDIX A (INFORMATIVE): SERVICE PRIMITIVES AND THE BLUETOOTH PDUs

In this Annex, we relate the service primitives shown in section 4.2 with the various Bluetooth PDUs which support these primitives. The table below only shows the actions taken at the higher involved Bluetooth layer. Thus, unless specifically stated, the low-level inquiries and pages needed to discover and connect to Bluetooth devices are not discussed in detail.

service primitive	(highest layer) Bluetooth PDUs involved
<p>serviceBrowse (LIST(<i>RemDev</i>) LIST(<i>RemDevRelation</i>) LIST(<i>browseGroup</i>) <i>getRemDevName</i> <i>stopRule</i>)</p>	<p>For the subset of <i>RemDev</i> that satisfy the <i>RemDevRelation</i>, this service primitive will cause the LocDev to send:</p> <p style="padding-left: 40px;">an <i>SDP_ServiceSearchRequest</i> PDU and receives a corresponding response PDU, see section 4.5 in BT_SDP_spec:[7];</p> <p style="padding-left: 40px;">an <i>SDP_ServiceAttributeRequest</i> PDU and receives a corresponding response PDU, see section 4.6 in BT_SDP_spec:[7].</p> <p>The first transaction above identifies the SDP servers that contain pertinent service records, while the second transaction retrieves the desired information;</p> <p>Alternatively, the two transactions above are combined to one:</p> <p style="padding-left: 40px;">LocDev sends an <i>SDP_ServiceSearchAttributeRequest</i> PDU and receives a corresponding response PDU, see section 4.7 in BT_SDP_spec:[7]</p> <p>In either of the above cases, the corresponding SDP transaction may last a number of request and response PDU exchanges, due to the L2CAP MTU limitation.</p> <p>If the <i>getRemDevName</i> parameter is set to 'yes', then for each <i>RemDev</i> involved in the execution of this service primitive, the service primitive will cause a sequence of <i>LMP_name_request()</i> LM level PDUs to be sent by the LocDev.* The corresponding <i>RemDev</i> responds with a <i>LMP_name_response()</i> LM level PDU containing the requested user-friendly device name.</p>
<p>serviceSearch (LIST(<i>RemDev</i>) LIST(<i>RemDevRelation</i>) LIST(<i>searchPattern</i>, <i>attributeList</i>) <i>getRemDevName</i> <i>stopRule</i>)</p>	<p>same as above</p>

Table 11.1: Bluetooth PDUs related to the service primitives in Section 4.2



service primitive	(highest layer) Bluetooth PDUs involved
enumerateRemDev (LIST(<i>classOfDevice</i>) <i>stopRule</i>)	This service primitive will cause a Bluetooth baseband <i>inquiry</i> process. The inquiry will ‘indiscriminately’ [†] find devices residing in the vicinity of the LocDev. Prior to returning the results of this inquiry the LocDev may filter them using the <i>classOfDevice</i> qualifier.
terminatePrimitive (<i>primitiveHandle</i> <i>returnResults</i>)	This service primitive will cause the termination of any outstanding operation caused by the invocation of the service primitive identified by the <i>primitiveHandle</i> parameter. This may cause an L2CAP connection termination request PDU to be sent from the LocDev to the RemDev, and the subsequent transmission of an L2CAP termination response PDU. If the LocDev is connecting to the RemDev only for the purposes of an SDP transaction, the baseband link will also be severed by the transmission of an LMP_detach LM level PDU.

Table 11.1: Bluetooth PDUs related to the service primitives in Section 4.2

- *. If the information requested is already stored (cached) in the LocDev, this service primitive may not have to cause the described LM level PDU transaction.
- †. The inquiries considered here use the GIAC. No CoD-specific DIACs have been defined. Nevertheless, the use of appropriate DIACs whenever possible is not excluded and is not outside the scope of this profile.

