Object Transfer Service

Bluetooth® Service Specification

- Date 2015-November-17
- Revision v10
- Group Prepared By Sports and Fitness WG
- Feedback Email sf-main@bluetooth.org

Abstract:

This service provides management and control features supporting bulk data transfers which occur via a separate L2CAP connection oriented channel. The Client is enabled to create and delete objects and to execute an action using the currently selected object. The selected object can be written, updated, or read via an Object Transfer Channel opened by the Client. The generation of a checksum covering a part or the whole of the object contents is included as an optional feature.

This service provides a general method for a Client to select and initiate the transfer of any type of object.

Revision History

Revision Number	Date (yyyy-mm-dd)	Comments
v10	2015-11-17	Adopted by the Bluetooth SIG BoD

Contributors

Name	Company
Robin Heydon	Qualcomm Technologies, Inc.
Tim Howes	Qualcomm Technologies, Inc.
Laurence Richardson	Qualcomm Technologies, Inc.
Guillaume Schatz	Polar Electro Oy
Robert D. Hughes	Intel Corporation
Ryan Moriyama	Beats Electronics
Victor Zhodzishsky	Broadcom Corporation
Leif-Alexandre Aschehoug	Nordic Semiconductor
Navin Kochar	Intel Corporation
Ella Chu	ISSC Technologies Corporation

DISCLAIMER AND COPYRIGHT NOTICE

This disclaimer applies to all draft specifications and final specifications adopted by the Bluetooth SIG Board of Directors (both of which are hereinafter referred to herein as a Bluetooth "Specification"). Your use of this Specification in any way is subject to your compliance with all conditions of such use, and your acceptance of all disclaimers and limitations as to such use, contained in this Specification. Any user of this Specification is advised to seek appropriate legal, engineering or other professional advice regarding the use, interpretation or effect of this Specification on any matters discussed in this Specification.

Use of Bluetooth Specifications and any related intellectual property is governed by the Promoters Membership Agreement among the Promoter Members and Bluetooth SIG (the "Promoters Agreement"), certain membership agreements between Bluetooth SIG and its Adopter and Associate Members, including, but not limited to, the Membership Application, the Bluetooth Patent/Copyright License Agreement and the Bluetooth Trademark License Agreement (collectively, the "Membership Agreements") and the Bluetooth Specification Early Adopters Agreements (1.2 Early Adopters Agreements) among Early Adopter members of the unincorporated Bluetooth SIG and the Promoter Members (the "Early Adopters Agreement"). Certain rights and obligations of the Promoter Members under the Early Adopters Agreements have been assigned to Bluetooth SIG by the Promoter Members.

Use of the Specification by anyone who is not a member of Bluetooth SIG or a party to an Early Adopters Agreement (each such person or party, a "Member") is prohibited. The use of any portion of a Bluetooth Specification may involve the use of intellectual property rights ("IPR"), including pending or issued patents, or copyrights or other rights. Bluetooth SIG has made no search or investigation for such rights and disclaims any undertaking or duty to do so. The legal rights and obligations of each Member are governed by the applicable Membership Agreements, Early Adopters Agreement or Promoters Agreement. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of the applicable Membership Agreements, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in (i) termination of the applicable Membership Agreements or Early Adopters Agreement and (ii) liability claims by Bluetooth SIG or any of its Members for patent, copyright and/or trademark infringement claims permitted by the applicable agreement or by applicable law.

THE SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the Bluetooth wireless technology ("Bluetooth Products") may be subject to various regulatory controls under the laws and regulations applicable to products using wireless non licensed spectrum of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of Bluetooth Products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their Bluetooth Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Bluetooth Products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses. NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. To the extent not prohibited by law, in no event will Bluetooth SIG or its Members or their affiliates be liable for any damages, including without limitation, lost revenue, profits, data or programs, or business interruption, or for special, indirect, consequential, incidental or punitive damages, however caused and regardless of the theory of liability, arising out of or related to any furnishing, practicing, modifying, use or the performance or implementation of the contents of this Specification, even if Bluetooth SIG or its Members or their affiliates have been advised of the possibility of such damages. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST BLUETOOTH SIG AND ITS MEMBERS OR THEIR AFFILATES RELATED TO USE OF THE SPECIFICATION.

If this Specification is an intermediate draft, it is for comment only. No products should be designed based on it except solely to verify the prototyping specification at SIG sponsored IOP events and it does not represent any commitment to release or implement any portion of the intermediate draft, which may be withdrawn, modified, or replaced at any time in the adopted Specification.

Bluetooth SIG reserves the right to adopt any changes or alterations to the Specification it deems necessary or appropriate.

Document Terminology

The Bluetooth SIG has adopted portions of the IEEE Standards Style Manual, which dictates use of the words “shall”, “should”, “may”, and “can” in the development of documentation, as follows:

The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*).

The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.

The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.

The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted*).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

The term *Reserved for Future Use (RFU)* is used to indicate Bluetooth SIG assigned values that are reserved by the Bluetooth SIG and are not otherwise available for use by implementations.

v10

Contents

v10

v10

v10

1 Introduction

The Object Transfer Service (OTS) requires the presence of L2CAP, GAP, SM (for LE), SDP (for BR/EDR), ATT, and GATT. This is illustrated in Figure 1.1. “L2CAP CoC” in this diagram denotes an L2CAP connection-oriented channel, hereinafter referred to as the “Object Transfer Channel”. The transfer of objects as bulk data is achieved by instructing the Server to send or receive data through the Object Transfer Channel.

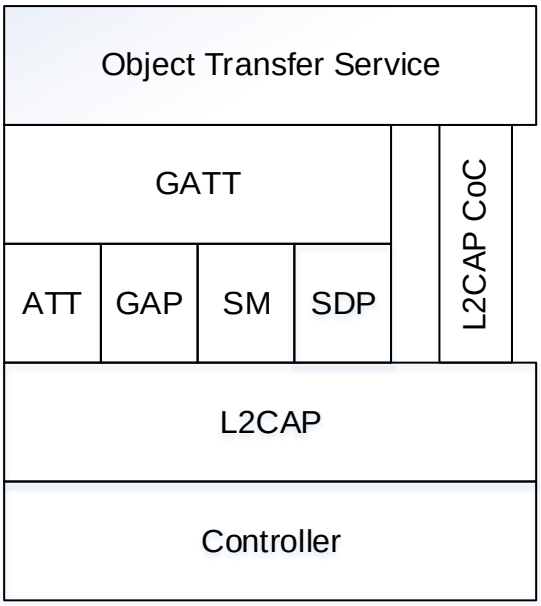

Figure 1.1: *Object Transfer Service Model*

Clients are enabled to create and delete objects and to execute an action on the Server device using the currently selected object. The selected object can be written, updated, or read via an Object Transfer Channel opened by the Client and hence transferred from Client to Server or from Server to Client. The generation of a checksum covering a part or the whole of the object contents is included as an optional feature.

The Server exposes the metadata of a single, selected object. Since there may be many objects available on a Server, this service provides features enabling the Client to filter objects based on their metadata and to control the selection of the ‘Current Object’ whose metadata will then be exposed to the Client via the Object Metadata characteristics defined in this specification.

A feature enabling objects to acquire a lock is included and this supports scenarios in which two or more Clients are connected and are using the same instance of the Object Transfer Service concurrently.

1.1 Conformance

If a device claims conformance to this service, all capabilities indicated as mandatory for this service 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 Qualification Program.

1.2 Service Dependency

This service is not dependent upon any other services.

1.3 Bluetooth Core Specification Release Compatibility

This specification is compatible with Bluetooth Core Specification v4.0 [1] or later.

1.4 GATT Sub-Procedure Requirements

Requirements in this section represent a minimum set of requirements for a server. Other GATT sub-procedures may be used if supported by both Client and Server.

Table 1.1 summarizes additional GATT sub-procedures required beyond those required by all GATT Servers.

GATT Sub-Procedure	Requirements
Write Characteristic Value	M
Read Characteristic Descriptor	M
Write Characteristic Descriptor	M
Indication	M
Read Long Characteristic Value	C.1
Write Long Characteristic Value	C.2

Table 1.1: *Additional GATT sub-procedure requirements*

C.1: Mandatory if the Server supports the GATT Write Long Characteristic Value sub-procedure or if the Server supports a readable characteristic value that may exceed (ATT_MTU-1) octets in length by other means, where ATT_MTU is the default size for the transport being used; optional otherwise.

C.2: Mandatory if the Server supports a characteristic value that may exceed (ATT_MTU-3) octets in length, where ATT_MTU is the default size for the transport being used, and the characteristic is writable; optional otherwise.

1.5 Transport Dependencies

This service has no transport dependency.

However, there is a direct relationship between the transport used by this GATT service and the transport to be used for performing the object transfer, as described in Sections 3.3.2.5 and 3.3.2.6.

Where the term BR/EDR is used throughout this document, this also includes the use of AMP.

1.6 Object Transfer Channel Description

The Object Transfer Channel is an L2CAP connection oriented channel that is suitable for bulk data transfer. The Object Transfer Channel may be supported over the LE transport or the BR/EDR transport.

It is always the responsibility of the Client to open the Object Transfer Channel, regardless of the direction of transfer, and is never the responsibility of the Server. The Object Transfer Channel may be closed either by the Server as described in Section 3.3.2.6.3 and Section 3.8 or, otherwise, by the Client.

1.7 Current Object

Where the term “Current Object” is used in this specification, it has the meaning given in this section.

The Object Metadata characteristics expose the metadata of only one object. In general, there may be many objects available on the Server and the Client can select one of these objects by using the procedures of the OLCP or, in the case of creating a new object, the procedures of the OACP:

- If the OLCP is used to select a different object and the operation is successful, the *newly selected* object becomes the Current Object as described in Section 3.4.
- If the OACP is used to create a new object and the operation is successful, the *newly created* object becomes the Current Object as described in Section 3.3.2.1.

The object whose metadata has been selected to be exposed via the Object Metadata characteristics is referred to as the “Current Object”.

If the Server implementation is capable of storing only a single object, that object is automatically the Current Object.

There is no requirement in this service for persistence of the selections made by the Client following disconnection. The next time the Client connects to the Server, the Current Object may be a different object (unless the Server is capable of storing only a single object). It is the responsibility of the Client to re-select the object it requires upon reconnection.

v10

1.8 Invalid Object

Where the term “Invalid Object” is used in this specification, it has the meaning given in this section.

The Current Object shall be deemed invalid if:

(i) the object has been deleted and another object has not yet been selected to become the new Current Object; or
(ii) filter conditions are set that cause the Current Object to be excluded from the list of objects.

A Current Object that became an Invalid Object due to the setting of filter conditions that caused the Current Object to be excluded from the list of objects ceases to be an Invalid Object if the filter conditions are changed again such that the Current Object is no longer excluded.

1.9 Object Contents

Where the term “object contents” is used in this specification it means the data forming the body of the object, excluding the object metadata. The object contents are transferred via the Object Transfer Channel whereas the object metadata are managed via GATT characteristics as described in Section 3.2.

The Object Transfer Service is indifferent to the internal structure of the object. Objects may be of any format, such as binaries, images, log files, XML, GPX, MIME etc. The specific Object Type is identified by a UUID as described in Section 3.2.3.

1.10 Application Error Codes

This service defines the following Attribute Protocol Application Error codes:

Name	Error Code	Description
Write Request Rejected	0x80	An attempt was made to write a value that is invalid or not supported by this Server for a reason other than the attribute permissions.
Object Not Selected	0x81	An attempt was made to read or write to an Object Metadata characteristic while the Current Object was an Invalid Object (see Section 1.8).
Concurrency Limit Exceeded	0x82	The Server is unable to service the Read Request or Write Request because it exceeds the concurrency limit of the service.
Object Name Already Exists	0x83	The requested object name was rejected because the name was already in use by an existing object on the Server.

Table 1.2: *Attribute Protocol Application Error codes defined by this service*

1.11 Byte Transmission Order

All characteristics used with this service shall be transmitted with the least significant octet first (i.e., little endian). The least significant octet is identified in this specification as well as in the characteristic definitions in [3].

1.11.1 Transmission of Strings Encoded in UTF-8

The transmission order of characters in strings and of the octets within multi-byte UTF-8 characters, such as within the Object Name characteristic, is defined in [3].

1.12 Concurrency

Servers that support two or more concurrent connections shall support the Concurrency Feature and shall behave as described in this section.

The maximum number of concurrently connected Clients that the Server is able to support for the purposes of this service shall be an integer represented by the term “MAX_CONCURRENCY” in this document.

1.12.1 Clients within the Concurrency Pool

The concurrently connected Clients being serviced in accordance with this sub-section are referred to in this document as the Clients in the “Concurrency Pool”.

When the number of concurrently connected Clients is less than or equal to MAX_CONCURRENCY, the Server shall support each Client as follows:

- The Server shall permit each Client to select an object independently of the selections made by the other Client(s), (i.e., the object selected by one Client may be the same as or different from the object selected by another Client). Thus, if one Client uses the Object List Control Point (OLCP) to

select an object to become its Current Object, this shall not change which object is selected as the Current Object for the purposes of any other Client.

- The values exposed via the Object Metadata characteristics and the Object List Filter characteristics, if supported, shall be specific to the Client. Therefore, if the Clients have selected different objects concurrently, the Server shall expose different object metadata values to each Client accordingly.
- If a Client writes a new value to an Object List Filter characteristic, as described in Section 3.5, it shall not affect the characteristic value that is exposed to another Client.

If the Server supports Object Action Control Point (OACP) procedures that enable the reading or writing of data to an object, refer to the following sections for additional requirements concerning the management of concurrent connections: the OACP Write procedure (Section 3.3.2.6), the OACP Abort procedure (Section 3.3.2.7), the Procedure Following Disconnection (Section 3.7), and the Object Transfer Timeout Procedure (Section 3.8).

1.12.2 Concurrency Limit Exceeded

If the number of concurrently connected Clients in the Concurrency Pool described in Section 1.12.1 is equal to MAX_CONCURRENCY and an additional connection is then established to a further device, the Server shall reject any requests from the newly connected device to read or write to the characteristics or characteristic descriptors of the Object Transfer Service. The Server shall send the Attribute Protocol Application Error code, “Concurrency Limit Exceeded”, defined in Section 1.10, to that Client in response to each such Write Request or Read Request.

This mechanism shall be used to deny access to the characteristics of the Object Transfer Service to all additional devices that connect while the number of Clients in the Concurrency Pool is equal to MAX_CONCURRENCY.

If a device that was within the Concurrency Pool later disconnects from the Server, the Server may allow a new device to join the Concurrency Pool. For example, if a device that was excluded from the Concurrency Pool has remained connected and attempts to read or write the characteristics of the Object Transfer Service, the Server may transfer it to the Concurrency Pool. Once a device has been transferred to the Concurrency Pool, the Server shall support it as described in Section 1.12.1.

2 Service Declaration

The Object Transfer Service (OTS) may be instantiated as a «Primary Service» or as a «Secondary Service». It is recommended that this service be instantiated as a «Secondary Service» which is referenced from the relevant primary service on the device. The primary service would provide the context for the object transfer and therefore the class of data being transferred. 'Class of data' here means one or more types of object that are of relevance to a particular service; object types are uniquely identified by UUID as described in Section 3.2.3.

The determination of whether a service is either a primary or secondary service can be mandated by a higher-level specification.

Multiple instances of the Object Transfer Service may be instantiated. For example, different primary services on the device may each include an instance of this service.

The service UUID shall be set to «Object Transfer Service» as defined in [3].

3 Service Characteristics

The characteristic requirements for an instance of the Object Transfer Service are shown in Table 3.1. With the exception of the Object List Filter characteristics (of which there shall be either three instances or none), only one instance of each characteristic is permitted within this service.

Each characteristic used by this service is defined in [3]. Information on the behavior of each characteristic is given in this section.

Characteristic Name	Requirement	Mandatory Properties	Optional Properties	Security Permissions
OTS Feature (Section 3.1)	M	Read	None	Encryption required
Object Name (Section 3.2.2)	M	Read	Write (C.3)	Encryption required
Object Type (Section 3.2.3)	M	Read	None	Encryption required
Object Size (Section 3.2.4)	M	Read	None	Encryption Required
Object First-Created (Section 3.2.5)	O	Read	Write (C.3)	Encryption Required
Object Last-Modified (Section 3.2.6)	O	Read	Write (C.4)	Encryption Required
Object ID (Section 3.2.7)	C.1	Read	None	Encryption Required
Object Properties (Section 3.2.8)	M	Read	Write	Encryption Required
Object Action Control Point (Section 3.3)	M	Write, Indicate	None	Encryption Required
Object List Control Point (Section 3.4)	C.1	Write, Indicate	None	Encryption Required
Object List Filter (Section 3.5)	C.2	Write, Read	None	Encryption Required
Object Changed (Section 3.6)	O	Indicate	None	Encryption Required

Table 3.1: *Requirements for characteristics used in the Object Transfer Service*

C.1: Mandatory if the Server is capable of storing more than one object; excluded otherwise.

C.2: Optional if the Server is capable of storing more than one object; excluded otherwise.

v10

C.3: Mandatory if the OACP Create op code is supported; optional otherwise.

C.4: Mandatory if the Server does not have access to a real time clock; excluded otherwise.

Notes:

1. Properties not listed as Mandatory or Optional are excluded for this version of the service.
2. Security Permissions of “None” means that this version of the service does not impose any requirement.
3. The characteristics and the characteristic descriptors specified by this service are defined in this specification and in [3].

For characteristics that may be indicated, a Client Characteristic Configuration descriptor shall be included in that characteristic as required by the Core Specification [1].

3.1 OTS Feature Characteristic

The OTS Feature characteristic exposes which optional features are supported by the Server implementation.

The structure of the characteristic is defined below:

LSO MSO

	OACP Features	OLCP Features
Octet Order	LSO...MSO	LSO...MSO
Data type	32bit	32bit
Size	4 octets	4 octets
Units	None	None

Table 3.2: *Structure of the OTS Feature characteristic*

All Reserved for Future Use (RFU) bits in the OTS Feature characteristic value shall be set to 0.

3.1.1 Characteristic Behavior

When read, the OTS Feature characteristic returns a value containing two fields, ‘OACP Features’ and ‘OLCP Features’. Each field is a bit field that may be used by a Client to determine certain supported features of the Server as defined below.

The value of the OTS Feature characteristic shall be static for the lifetime of the device (i.e., static permanently or until Service Changed is indicated).

3.1.1.1 OACP Features Field

When a bit is set to 1 (True) in the OACP Features field, the Server supports the associated operation as defined in the following table:

Bit Number	Definition
0	OACP Create Op Code Supported 0 = False 1 = True
1	OACP Delete Op Code Supported 0 = False 1 = True
2	OACP Calculate Checksum Op Code Supported 0 = False 1 = True
3	OACP Execute Op Code Supported 0 = False 1 = True
4	OACP Read Op Code Supported 0 = False 1 = True
5	OACP Write Op Code Supported 0 = False 1 = True
6	Appending Additional Data to Objects Supported 0 = False 1 = True
7	Truncation of Objects Supported 0 = False 1 = True
8	Patching of Objects Supported 0 = False 1 = True
9	OACP Abort Op Code Supported 0 = False 1 = True
10-31	Reserved for Future Use

Table 3.3: *Definition of the bits of the OACP Features field*

If the Server does not support the relevant operation, the associated feature bit shall be set to 0 (False).

If the Server does not support the Object Action Control Point, all bits of the OACP Features field shall be set to 0.

3.1.1.2 OLCP Features Field

When a bit is set to 1 (True) in the OLCP Features field, the Server supports the associated operation as defined in the following table:

Bit Number	Definition
0	OLCP *Go To* Op Code Supported 0 = False 1 = True
1	OLCP *Order* Op Code Supported 0 = False 1 = True
2	OLCP *Request Number of Objects* Op Code Supported 0 = False 1 = True
3	OLCP *Clear Marking* Op Code Supported 0 = False 1 = True
4-31	Reserved for Future Use

Table 3.4: *Definition of the bits of the OLCP Features field*

If the Server does not support the relevant operation, the associated feature bit shall be set to 0 (False).

If the Server does not support the Object List Control Point, all bits of the OLCP Features field shall be set to 0.

3.2 Object Metadata Characteristics

The Object Metadata characteristics are a group of characteristics that expose the metadata of the Current Object. The values exposed by the Object Metadata characteristics shall be updated by the Server whenever the selection of the Current Object changes as a result of use of the Object List Control Point (OLCP) or Object Action Control Point (OACP) as described in Section 3.2.1. Thus, the values of the Object Metadata characteristics may change during the execution of a control point procedure and are only considered valid once the corresponding control point response has been indicated.

Whether support for each characteristic is mandatory, optional, or conditional has been defined in Table 3.1. For a given implementation of this service, the same set of Object Metadata characteristics shall be supported for all the supported objects and the characteristics shall be populated with the values for the Current Object. For the date-based object metadata, the sub-sections below also define special values for use when it is necessary to indicate that a valid date is not available in respect of the Current Object.

For each Object Metadata characteristic defined in the sub-sections below, if the characteristic has attribute write permissions, the Server may choose whether to reject a write request to that characteristic if it determines that the contents of the new value are unsuitable, such as a string containing characters in a language that the implementation does not support. The criteria by which a requested value may be

determined to be unsuitable are left to the implementation. The Server may reject such a write request by sending the applicable Attribute Protocol Application Error code “Write Request Rejected” as defined in Section 1.10.

The conditions under which the Current Object becomes an Invalid Object are described in Section 1.8. When the Current Object is an Invalid Object, any attempt to read any of the Object Metadata characteristics shall be rejected by responding with the Attribute Protocol Application Error code “Object Not Selected” as defined in Section 1.10. Similarly, when the Current Object is an Invalid Object and an Object Metadata characteristic has attribute write permissions, any attempt to write to that Object Metadata characteristic shall also be rejected by responding with the Attribute Protocol Application Error code “Object Not Selected”.

3.2.1 Selection of the Current Object

The object whose metadata are to be exposed via the Object Metadata characteristics (i.e., the Current Object) is selected as described in Section 1.7.

3.2.2 Object Name Characteristic

The Object Name characteristic exposes the name of the Current Object.

3.2.2.1 Object Name Characteristic Behavior

The Object Name characteristic returns its associated value when read.

If the Object Name characteristic is writable, the Server shall allow new values to be written to this characteristic by a Client. However, if the Client attempts to write a name which already belongs to another object on the Server, the Server shall reject such a write request by sending the applicable Attribute Protocol Application Error code “Object Name Already Exists” as defined in Section 1.10.

Whenever a value is successfully written to the characteristic, the Server shall also update its local record of the object’s name metadata and retain the new value for that object.

Refer to the OACP Create procedure in Section 3.3.2.1 for additional requirements.

3.2.3 Object Type Characteristic

This characteristic exposes the type of the Current Object, identifying the object type by UUID.

Relevant object types that are to be used for a given application may be specified by a higher-level specification.

3.2.3.1 Object Type Characteristic Behavior

The Object Type characteristic returns its associated value when read.

The value of the characteristic is of variable length because it contains either a 16-bit or 128-bit UUID that identifies the type of the Current Object. The Client can detect whether a UUID using the 16-bit format (2 octets) or 128-bit format (16 octets) has been received by checking the length of the Object Type characteristic value.

UUIDs that use the 16-bit format are defined in the Bluetooth SIG Assigned Numbers [3].

Refer to the OACP Create procedure in Section 3.3.2.1 for additional requirements.

3.2.4 Object Size Characteristic

The Object Size characteristic exposes the current size as well as the allocated size of the Current Object.

The structure of the characteristic is defined below:

LSO MSO

	Current Size	Allocated Size
Octet Order	LSO...MSO	LSO...MSO
Data type	UINT32	UINT32
Size	4 octets	4 octets
Units	unitless	unitless

Table 3.5: *Structure of the Object Size characteristic*

3.2.4.1 Object Size Characteristic Behavior

When read, the Object Size characteristic returns a value containing two fields: 'Current Size' and 'Allocated Size', as described below.

3.2.4.1.1 Current Size Field

The value of the Current Size field represents an integer number of octets equal to the actual size of the object. The value of Current Size shall be less than or equal to the value of Allocated Size.

After data has been written to an object as the result of an OACP Write operation, the value of Current Size shall be updated to report the new size of the object. As the result of object data being written successfully, the size of the object usually changes (the one exception to this being when existing data is overwritten without affecting the object's size).

Note: In the event that such an object transfer becomes terminated before the transfer is completed (e.g., due to link loss), the updated value of the Current Size field allows the Client to resume the object transfer efficiently by re-starting from the octet position following the last octet that was successfully written. The octet numbering is based from zero; i.e., the first octet of the object contents has an index of zero, the second octet has an index of one, etc. Using the updated value of the Current Size field as the value for the Offset parameter of the OACP Write procedure will therefore resume the object transfer from the required position. This resumption method can be used provided that the value of Current Size has changed; it cannot be used in the case where existing data has been overwritten without affecting the object's size.

Refer to the OACP Write procedure described in Section 3.3.2.6 for additional requirements.

3.2.4.1.2 Allocated Size Field

The value of this field is equal to the number of octets that have been allocated by the Server for the object contents. This value indicates the amount of memory the Server has reserved for holding the object contents.

Refer to the OACP Create procedure in Section 3.3.2.1 for the role of the *Size* parameter in setting the initial value for the allocated size when a new object is created via that procedure.

In the event that additional data is successfully appended to the object such that its size would be greater than the present value of the Allocated Size field, the Server shall increase the Allocated Size to accommodate the extended object. Thus, the value of the Allocated Size field shall be greater than or equal to the value of the Current Size field.

3.2.5 Object First-Created Characteristic

The Object First-Created characteristic exposes a value representing a date and time when the object contents were first created. Note that this may be an earlier date than the date of the creation of the object on the present Server; it can relate to the original file creation date of a firmware release, for instance, rather than the time that the object was written to the Server.

The date and time value shall be represented in Coordinated Universal Time (UTC).

Note that, once the value of the object first-created metadata has been set to a valid date and time (i.e., populated with non-zero values), its value is not intended to change any further, since the time of first creation of an object is a constant by definition.

3.2.5.1 Object First-Created Characteristic Behavior

The Object First-Created characteristic returns its associated value when read.

The Server may allow a value to be written to this characteristic. When a value is successfully written to the characteristic, the Server shall also update its internal record of the object's first-created metadata and retain the new value for that object.

If an object is created at the Server and the Server does not have access to a valid date and time to set the object first-created metadata, the fields of the object first-created metadata for which the Server does not have valid data shall be set to 0.

Refer to the OACP Create procedure in Section 3.3.2.1 for additional requirements.

3.2.6 Object Last-Modified Characteristic

The Object Last-Modified characteristic exposes a value representing a date and time when the object content was last modified.

The date and time value shall be represented in Coordinated Universal Time (UTC).

3.2.6.1 Object Last-Modified Characteristic Behavior

The Object Last-Modified characteristic returns its associated value when read.

If the Server has access to a real time clock, the Object Last-Modified characteristic shall not be writable and the Server shall update the object last-modified metadata itself when a relevant event occurs on the Server, as determined by the implementation.

If the Server does not have access to a real time clock, the Object Last-Modified characteristic shall be writable. When a value is successfully written to the characteristic, the Server shall also update its internal record of the object's last-modified metadata and retain the new value for that object.

When an object is modified and the Server does not have access to a real time clock, the fields of the object last-modified metadata for which the Server does not have valid data shall be reset to 0.

If an object is created at the Server and the Server does not have access to a valid date and time to set the object last-modified metadata, the fields of the object last-modified metadata for which the Server does not have valid data shall be set to 0.

Refer to the OACP Create procedure in Section 3.3.2.1 for additional requirements.

3.2.7 Object ID Characteristic

The Object ID characteristic exposes an integer value which is the Object ID of the Current Object. The Object ID is a LUID (Locally Unique Identifier).

The allocated Object ID shall be unique on the Server and shall be a UINT48 value in the range 0x000000000100 to 0xFFFFFFFFFFFF.

The value 0x000000000000 is reserved for the Directory Listing Object as described in Section 4.1.

The values 0x000000000001 to 0x0000000000FF are reserved for future use.

3.2.7.1 Object ID Characteristic Behavior

The Object ID characteristic returns its associated value when read.

The Server shall allocate an Object ID value to each object when the object is created on the Server as described in Section 3.3.2.1. This value shall be unique on the Server. The values allocated are not required to be sequential numbers.

Once an object has been successfully deleted, its Object ID may be returned to the pool for re-use. However, Object IDs that have not been previously used should be allocated prior to re-using released Object IDs.

Refer to the OACP Create procedure in Section 3.3.2.1 for additional requirements.

3.2.8 Object Properties Characteristic

This characteristic exposes a bit field representing the properties of the Current Object. When a bit is set to 1 (True), the Current Object possesses the associated property as defined in [3].

An object's properties determine the operations that are permitted to be performed on that object, as described in Section 3.3. For example, an attempt to use the OACP Write Op Code when the Write bit of the properties for the Current Object is 0 (False) will result in an error response.

	Object Properties
Octet Order	LSO … MSO
Data type	32bits
Size	4 octets

***Table 3.6**: Format of the Object Properties Characteristic*

When a bit of the Object Properties characteristic is set to 1 (i.e. True), the Current Object possesses the associated property as defined in the following table:

Bit	Property	Description
0	Delete	Deletion of this object is permitted: 0: False 1: True
1	Execute	Execution of this object is permitted: 0: False 1: True
2	Read	Reading this object is permitted: 0: False 1: True
3	Write	Writing data to this object is permitted: 0: False 1: True
4	Append	Appending data to this object that increases its Allocated Size is permitted: 0: False 1: True
5	Truncate	Truncation of this object is permitted: 0: False 1: True
6	Patch	Patching this object by overwriting some of the object's existing contents is permitted: 0: False 1: True
7	Mark	This object is a marked object: 0: False 1: True
8 - 31	RFU	Reserved for Future Use.

***Table 3.7**: Object Properties Definition*

3.2.8.1 Object Properties Characteristic Behavior

The Object Properties characteristic returns its associated value when read.

If this characteristic is writable, the Server may reject a requested change to the properties of the Current Object by sending the Attribute Protocol Application Error code "Write Request Rejected" as defined in Section 1.10. However, if a value is successfully written to the characteristic, the Server shall update its internal record of the object and retain the new properties for that object.

If the Server supports the Concurrency Feature and more than one Client has selected the same object concurrently, the Server may use the above mechanism to reject a write request to the Object Properties characteristic or it may accept such write requests; this is left to the implementation.

Refer to the OACP Create procedure in Section 3.3.2.1 for additional requirements.

3.3 Object Action Control Point (OACP)

The structure of the OACP characteristic is defined below:

LSO MSO

	Op Code	Parameter
	(See also Table 3.9.)	
Octet Order	N/A	LSO...MSO
Data type	UINT8	Variable
Size	1 octet	0 to 20 octets
Units	None	None

Table 3.8: *Structure of the Object Action Control Point*

3.3.1 Object Action Control Point Procedure Requirements

Table 3.9 shows the requirements for the OACP characteristic. The Server shall support at least one OACP procedure.

v10

Op Code Value	Procedure	Requirement	Parameter	Applicable Response Values	Response Parameter
0x00	Reserved for Future Use				
0x01	Create	O	Size (UINT32), Type (gatt_uuid) See Note 1.	Refer to Section 3.3.2.1.	None
0x02	Delete	O	None	Refer to Section 3.3.2.2.	None
0x03	Calculate Checksum	O	Offset (UINT32), Length (UINT32)	Refer to Section 3.3.2.3.	Checksum (UINT32)

Op Code Value	Procedure	Requirement	Parameter	Applicable Response Values	Response Parameter
0x04	Execute	O	Parameter may be defined by a higher-level spec; none otherwise.	Refer to Section 3.3.2.4.	Response Parameter may be defined by a higher-level spec; none otherwise.
0x05	Read	O	Offset (UINT32), Length (UINT32)	Refer to Section 3.3.2.5.	None
0x06	Write	O	Offset (UINT32), Length (UINT32), Mode (8bit)	Refer to Section 3.3.2.6.	None
0x07	Abort	C.1	None	Refer to Section 3.3.2.7.	None
0x08-0x5F	Reserved for Future Use				
0x60	Response Code	M	OACP Response Value (see Table 3.10)	N/A	N/A
0x61-0xFF	Reserved for Future Use				

***Table 3.9**: Object Action Control Point Procedure Requirements*

C.1: Optional if the OACP Read procedure is supported; excluded otherwise.

Note 1: The gatt_uuid format type defines UUIDs of more than one size, including a large 128-bit UUID. This service supports all UUIDs that conform to the gatt_uuid format. However, note that it will only be possible to create objects with an Object Type represented by a 128-bit UUID via this control point if the Server is able to negotiate an ATT_MTU size with the Client that is 24 octets or larger or else by using the GATT Write Long Characteristic Values sub-procedure to write the value.

The OACP Response Value is of variable length and shall be formatted as follows:

LSO MSO

	OACP Response Value		
	Request Op Code	**Result Code**	**Response Parameter (if present)**
		(See also Table 3.11.)	
Octet Order	N/A	N/A	LSO…MSO
Data type	UINT8	UINT8	variable
Size	1 octet	1 octet	variable

Table 3.10: *Format of OACP Response Value*

The Result Codes associated with Op Code 0x60 are defined in Table 3.11:

Result Code	Definition	Description
0x00	Reserved For Future Use	
0x01	Success	Response for successful operation.
0x02	Op Code Not Supported	Response if unsupported Op Code is received.
0x03	Invalid Parameter	Response if Parameter received does not meet the requirements of the service.
0x04	Insufficient Resources	Response if the number of octets requested via the value of the Length parameter or Size parameter (as applicable) exceeds the available memory or processing capabilities of the Server.
0x05	Invalid Object	Response if the requested OACP procedure cannot be performed because the Current Object is an Invalid Object.
0x06	Channel Unavailable	Response if the requested procedure could not be performed because an Object Transfer Channel was not available for use.
0x07	Unsupported Type	Response if the object type specified in the OACP procedure Type parameter is not supported by the Server.
0x08	Procedure Not Permitted	Response if the requested procedure is not permitted according to the properties of the Current Object (refer to Section 3.2.8).
0x09	Object Locked	Response if the Current Object is temporarily locked by the Server.
0x0A	Operation Failed	Response if the requested procedure failed for any reason other than those enumerated in this table.
0x0B-0xFF	Reserved For Future Use	

Table 3.11: *List of OACP Result Codes*

3.3.2 Object Action Control Point Characteristic Behavior

The Object Action Control Point is used by a Client to control certain behaviors of the Server. With the exception of the Create procedure which creates a new object, the OACP procedures affect the Current Object only.

The procedures are triggered by writing a value that includes an Op Code specifying the operation and this may be followed by a Parameter that is valid within the context of that Op Code (see Table 3.9). For each of the procedures described in the next sections, the Server shall indicate the OACP characteristic with the Response Code Op Code (0x60) along with the Request Op Code and “Success” or other appropriate Result Code contained in the response as listed in Table 3.11.

Refer also to Section 3.3.2.8 for information on General Error Handling.

3.3.2.1 Create Procedure

When the *Create* Op Code is written to the OACP along with *Type* and *Size* parameters and an error condition does not occur, the Server shall create a new, empty object.

The object metadata values relating to the newly created object and hence the values of the Object Metadata characteristics shall be initialized as described in Section 3.3.2.1.1 below. After creating the new object, the Server shall respond with the “Success” result code.

However, if an error condition occurs, the “Success” result code shall not be sent and the appropriate error response shall be sent as defined in Table 3.11 and Table 3.12:

Error Condition	**Error Response** (See Section 3.3.2.8)
The Create Op Code is not supported by the Server.	Op Code Not Supported
The Server does not accept an object of the type specified in the Type parameter.	Unsupported Type
The Server cannot accept an object of the size specified in the Size parameter.	Insufficient Resources
The Parameter received does not meet the requirements of the service.	Invalid Parameter
The requested procedure failed for a reason other than those enumerated above in this table.	Operation Failed

Table 3.12: *Requirements for Control Point Error Responses to the Create Op Code*

In the event that more than one error condition listed in Table 3.12 occurs in the same transaction, the applicable Error Response which is nearest to the top of the table shall take priority and only this Error Response shall be sent.

If the response to the *Create* Op Code is “Success”, the value of the Object List Filter characteristics, if supported, shall be updated by the Server to ensure that the filter type is set to “No Filter” and the newly created object shall be designated as the Current Object.

However, if the response to the *Create* Op Code is not “Success”, the filter type and the object selected as the Current Object shall remain the same as they were prior to the Create procedure being attempted.

3.3.2.1.1 Initialization of Object Metadata

When a new object is successfully created:

- the name metadata of the object and the value reported in the Object Name characteristic shall be set to a zero length string.
- the type metadata of the object and the value reported in the Object Type characteristic shall be set to the value of the *Type* parameter included with the *Create* Op Code.
- the amount of memory reserved for the object contents and the value reported in the Allocated Size field of the Object Size characteristic shall be set to a value greater than or equal to the value of the *Size* parameter included with the *Create* Op Code.
- the value of the Current Size field of the Object Size characteristic shall be set to 0. This is because the object contents are initially empty.
- the value of all fields of the Object First-Created characteristic, if supported, shall be set to 0. This is a special value indicating that the date and/or time are not valid.
- the value of all fields of the Object Last-Modified characteristic, if supported, shall be set to 0. This is a special value indicating that the date and/or time are not valid.
- the value of the Object ID characteristic, if supported, shall be set to a value dynamically allocated to the object on its creation. Criteria governing the allocation of the value are defined in Section 3.2.7.
- the object’s properties and the value reported in the Object Properties characteristic shall be initialized. The Write property shall initially be set to True so that writing object contents to the empty object is enabled. The remaining properties of the object should be set as required by the implementation.

v10

3.3.2.2 Delete Procedure

When the *Delete* Op Code is written to the OACP and an error condition does not occur, the Server shall delete the Current Object. After deleting the object, the Server shall respond with the “Success” result code.

If the response to the *Delete* Op Code is “Success”, this operation shall make the Current Object invalid until a new object is selected to become the Current Object. This is necessary because the “pointer” to the Current Object is left pointing at something that has been deleted, until it is updated by selecting a different object. The status of the Current Object as valid or invalid is tracked internally and is left to the implementation.

Any attempt to use further OACP procedures on an Invalid Object following its successful deletion shall be rejected as described in Section 3.3.2.8.2.4.

However, if an error condition occurs, the “Success” result code shall not be sent and the appropriate error response shall be sent as defined in Table 3.11 and Table 3.13:

Error Condition	Error Response (See Section 3.3.2.8)
The *Delete* Op Code is not supported by the Server.	Op Code Not Supported
The Current Object is an Invalid Object.	Invalid Object
The object's properties do not permit deletion of the object.	Procedure Not Permitted
The Current Object is locked by the Server.	Object Locked
An object transfer is in progress that is using the Current Object.	Object Locked
The requested procedure failed for a reason other than those enumerated above in this table.	Operation Failed

Table 3.13: *Requirements for Control Point Error Responses to the Delete Op Code*

In the event that more than one error condition listed in Table 3.13 occurs in the same transaction, the applicable Error Response which is nearest to the top of the table shall take priority and only this Error Response shall be sent.

If the response to the *Delete* Op Code is not "Success", the object selected as the Current Object shall remain the same as it was prior to the Delete procedure being attempted.

3.3.2.3 Calculate Checksum Procedure

This procedure enables a Client to compare a checksum that it has calculated itself with a checksum provided by the Server for the same octets, so that the Client can detect any discrepancy in the data transferred in either direction.

Servers that support the OACP Calculate Checksum procedure shall support the generation of a checksum calculated from the whole or of any part of the Server-side object contents as follows:

An *Offset* parameter is required with the *Calculate Checksum* Op Code. This parameter is used to request a checksum covering the octets beginning from a specified octet position within the object. The value represents an integer number of octets and is based from zero; the first octet of the object contents has an index of zero, the second octet has an index of one, etc. The octet numbered zero is the least significant octet.

A *Length* parameter is required with the *Calculate Checksum* Op Code. The object contents included in the calculation shall consist of *Length* number of octets, starting with the octet identified by the value of the *Offset* parameter.

When the *Calculate Checksum* Op Code is written to the OACP and an error condition does not occur, the Server shall calculate a 32-bit CRC value from the specified octets of the Current Object, using an ISO/IEC 3309 compliant 32-bit CRC algorithm [4]. After calculating the checksum, the Server shall respond with the "Success" result code. The newly calculated checksum value shall be provided in the Checksum response parameter.

However, if an error condition occurs, the "Success" result code shall not be sent and the appropriate error response shall be sent as defined in Table 3.11 and Table 3.14:

Error Condition	Error Response (See Section 3.3.2.8)
The *Calculate Checksum* Op Code is not supported by the Server.	Op Code Not Supported
The Current Object is an Invalid Object.	Invalid Object
The sum of the values of the *Offset* and *Length* parameters exceeds the value of the Current Size field of the Object Size characteristic.	Invalid Parameter
The Current Object is locked by the Server.	Object Locked
An object transfer is in progress that is using the Current Object.	Object Locked
The requested procedure failed for a reason other than those enumerated above in this table.	Operation Failed

Table 3.14: *Requirements for Control Point Error Responses to the Calculate Checksum Op Code*

In the event that more than one error condition listed in Table 3.14 occurs in the same transaction, the applicable Error Response which is nearest to the top of the table shall take priority and only this Error Response shall be sent.

Although the Object Transfer Channel is highly reliable, the OACP Calculate Checksum procedure provides a further level of data integrity.

Note that an object may also include a signature, a hash, and/or other security features within it; use of such embedded features is outside the scope of this specification.

v10

3.3.2.4 Execute Procedure

When the *Execute* Op Code is written to the OACP and an error condition does not occur, the Server shall use the Current Object to perform an operation on the Server device. After initiating execution of the object, the Server shall respond with the "Success" result code. The nature of the operation initiated by receipt of the *Execute* Op Code will be determined by the Object Type definition. A typical context is the execution of a firmware update following transfer of a new firmware object to the Server and successful data integrity checks.

However, if an error condition occurs, the "Success" result code shall not be sent and the appropriate error response shall be sent as defined in Table 3.11 and Table 3.15:

Error Condition	Error Response (See Section 3.3.2.8)
The *Execute* Op Code is not supported by the Server.	Op Code Not Supported
The Current Object is an Invalid Object.	Invalid Object
The object's properties do not permit execution of the object.	Procedure Not Permitted
The Current Object is locked by the Server.	Object Locked
The requested procedure failed for a reason other than those enumerated above in this table.	Operation Failed

Table 3.15: *Requirements for Control Point Error Responses to the Execute Op Code*

In the event that more than one error condition listed in Table 3.15 occurs in the same transaction, the applicable Error Response which is nearest to the top of the table shall take priority and only this Error Response shall be sent.

For this procedure, receiving the “Success” response means only that the Server has accepted the request. The application-specific behavior that follows receipt of the *Execute* Op code is outside the scope of this document and may be defined in a higher-level specification, including any Parameter and/or Response Parameter that is required for use with the *Execute* Op code.

3.3.2.5 Read Procedure

When the *Read* Op Code is written to the OACP and an error condition does not occur, the Server shall send object data to the Client through the Object Transfer Channel opened by the Client. The data to be sent is read from the Current Object as described below.

When the *Read* Op Code is written to the OACP characteristic over the LE Transport, the Server shall consider this as a request to perform the object transfer over LE and shall require the Client to have opened the required Object Transfer Channel using the LE Transport. Conversely, if the relevant OACP Op Code was written to the OACP characteristic over the BR/EDR Transport, it shall consider this as a request to perform the object transfer over BR/EDR and shall require the Client to have opened the required Object Transfer Channel using the BR/EDR Transport.

For this procedure, “Success” is defined to mean that the Server has accepted the request and will commence sending the requested octets towards the Client through the open Object Transfer Channel. The Server shall respond with the “Success” result code as soon as it is ready to send the data through the Object Transfer Channel.

The total number of octets that shall be read from the Current Object is specified by the *Length* parameter.

The position of the first octet that shall be read from the Current Object is specified by the value of the *Offset* parameter. For example, if the value of the *Offset* parameter is 0x0003, the first octet to be read from the object and sent through the Object Transfer Channel shall be octet number 0x0003, as shown in Figure 3.1:

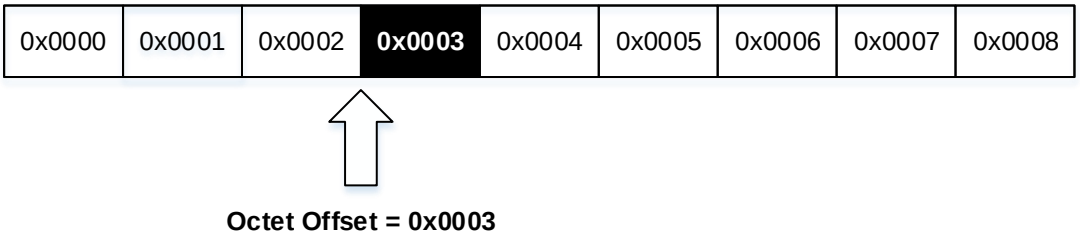

Figure 3.1: *Reading Data from an Offset Position*

However, if an error condition occurs, the “Success” result code shall not be sent and the appropriate error response shall be sent as defined in Table 3.11 and Table 3.16:

Error Condition	Error Response (See Section 3.3.2.8)
The *Read* Op Code is not supported by the Server.	Op Code Not Supported
The Current Object is an Invalid Object.	Invalid Object
The object's properties do not permit reading the object.	Procedure Not Permitted
An Object Transfer Channel was not available for use.	Channel Unavailable
The value of the *Offset* parameter exceeds the value of the Current Size field of the Object Size characteristic.	Invalid Parameter
The sum of the values of the *Offset* and *Length* parameters exceeds the value of the Current Size field of the Object Size characteristic.	Invalid Parameter
The value of the *Length* parameter exceeds the number of octets that the Server has the capacity to read from the object.	Insufficient Resources
An object transfer is already in progress that is using the Current Object.	Object Locked
The requested procedure failed for a reason other than those enumerated above in this table.	Operation Failed

Table 3.16: *Requirements for Control Point Error Responses to the Read Op Code*

In the event that more than one error condition listed in Table 3.16 occurs in the same transaction, the applicable Error Response which is nearest to the top of the table shall take priority and only this Error Response shall be sent.

If the OACP Abort procedure is supported and is executed, an object transfer that has been initiated via this OACP Read procedure and is still in progress via the Object Transfer Channel shall be aborted in accordance with the OACP Abort procedure described in Section 3.3.2.7.

Refer also to Section 3.8 for the Object Transfer Timeout procedure.

3.3.2.6 Write Procedure

When the *Write* Op Code is written to the OACP and an error condition does not occur, the Server shall open the Current Object for writing and prepare to accept data from the Client through the Object Transfer Channel opened by the Client, writing the data received to the Current Object.

When the *Write* Op Code is written to the OACP characteristic over the LE Transport, the Server shall consider this as a request to perform the object transfer over LE and shall require the Client to have opened the required Object Transfer Channel using the LE Transport. Conversely, if the relevant *OACP* Op Code was written to the OACP characteristic over the BR/EDR Transport, it shall consider this as a request to perform the object transfer over BR/EDR and shall require the Client to have opened the required Object Transfer Channel using the BR/EDR Transport.

For this procedure, "Success" is defined to mean that the Server is ready to accept the specified number of octets from the Client through the open Object Transfer Channel. The Server shall respond with the "Success" result code as soon as it is ready to receive the data through the Object Transfer Channel.

The total number of octets that shall be accepted is specified by the *Length* parameter.

The *Mode* parameter contains a bit field that specifies how the write operation shall be performed as defined in Table 3.17:

Mode Parameter		
Bit	**Definition**	**Description**
0	Reserved For Future Use	
1	Truncate	The object shall be truncated: 0: False 1: True
2-7	Reserved For Future Use (RFU)	

Table 3.17: *Mode Parameter for OACP Write Op Code*

If the Truncate bit of the *Mode* parameter is set to 0 (False), the object shall not be truncated. This mode allows any part of the object contents to be overwritten without truncating the object.

If the Truncate bit of the *Mode* parameter is set to 1 (True) and the Server responds with the “Success” result code, the object shall be truncated such that its Current Size becomes equal to the value of the *Offset* parameter before new data is written to the object. Once data has been written successfully to the object, the value of the Current Size field of the Object Size characteristic shall be updated again by the Server to represent the new Current Size. This mode allows the size of an object to change dynamically. The position of the first octet that shall be written in the Current Object is specified by the value of the *Offset* parameter. For example, if the value of the *Offset* parameter is 0x0003, the first octet to be written in the object when received through the Object Transfer Channel shall be octet number 0x0003, as shown in Figure 3.2:

Figure 3.2: *Writing Data to an Offset Position*

However, if an error condition occurs, the “Success” result code shall not be sent and the appropriate error response shall be sent as defined in Table 3.11 and Table 3.18:

Error Condition	**Error Response** (See Section 3.3.2.8)
The *Write* Op Code is not supported by the Server.	Op Code Not Supported
The Current Object is an Invalid Object.	Invalid Object
The object’s properties do not permit writing to the object.	Procedure Not Permitted

Error Condition	Error Response (See Section 3.3.2.8)
Patching was attempted but patching is not supported by the Server.	Procedure Not Permitted
Patching was attempted but the object's properties do not permit patching of the object contents.	Procedure Not Permitted
Truncation was attempted but the object's properties do not permit truncation of the object contents.	Procedure Not Permitted
An Object Transfer Channel was not available for use.	Channel Unavailable
The *Mode* parameter contains an RFU value.	Invalid Parameter
The value of the *Offset* parameter exceeds the value of the Current Size field of the Object Size characteristic.	Invalid Parameter
The sum of the values of the *Offset* and *Length* parameters exceeds the value of the Allocated Size field of the Object Size characteristic AND the Server does NOT support appending additional data to an object.	Invalid Parameter
The value of the *Length* parameter exceeds the number of octets that the Server has the capacity to write to the object.	Insufficient Resources
The Current Object is locked by the Server.	Object Locked
An object transfer is already in progress that is using the Current Object.	Object Locked
The requested procedure failed for a reason other than those enumerated above in this table.	Operation Failed

Table 3.18: *Requirements for Control Point Error Responses to the Write Op Code*

In the event that more than one error condition listed in Table 3.18 occurs in the same transaction, the applicable Error Response which is nearest to the top of the table shall take priority and only this Error Response shall be sent.

If the Server supports the Concurrency Feature described in Section 1.12, the Server shall prevent any other Client from writing data to the same object by temporarily locking the object until the object transfer has completed.

Refer also to Section 3.8 for the Object Transfer Timeout procedure.

If the Server supports appending additional data to an object, as indicated by the relevant bit of the OTS Feature characteristic, and the sum of the values of the *Offset* and *Length* parameters exceeds the value of the Allocated Size field of the Object Size characteristic, and none of the error conditions listed in Table 3.18 has occurred, the Server shall proceed as described in Section 3.3.2.6.1.

3.3.2.6.1 Increasing the Allocated Size of an Object

This sub-section applies only if the sum of the values of *Offset* and *Length* parameters exceeds the value of the Allocated Size field of the Object Size characteristic and none of the error conditions listed in Table

3.18 have occurred. The Server shall handle this as a request to increase the object's allocated size and the additional requirements of this sub-section shall apply.

If none of the error conditions listed in Table 3.19 applies, the Server shall update the value of the Allocated Size field of the Object Size characteristic to equal the sum of the values of *Offset* and *Length* parameters. The Server shall then proceed to accept data from the Client through the Object Transfer Channel as described in Section 3.3.2.6.

However, if any of the following additional error conditions occur, the "Success" result code shall not be sent and the appropriate error response shall be sent as defined in Table 3.11 and Table 3.19:

Error Condition	Error Response
The object's properties do not permit appending data to the object.	Procedure Not Permitted
The sum of the values of the *Offset* and *Length* parameters exceeds the capability of the Server to increase the object allocated size, owing to resource constraints.	Insufficient Resources

Table 3.19: *Supplementary Requirements for Control Point Error Responses to the Write Op Code that Apply to Appending Additional Data to an Object*

In the event that more than one error condition listed in Table 3.19 occurs in the same transaction, the applicable Error Response which is nearest to the top of the table shall take priority and only this Error Response shall be sent.

3.3.2.6.2 Overwriting Existing Data

Since values of the Offset and Length parameters may be chosen to select octets within the object that have already been written previously, this procedure enables a part or the whole of an object to be overwritten with new data. However, the Server may or may not allow patching; support is indicated by the relevant bit of the OTS Feature characteristic as described in Section 3.1.1.1.

Whether or not an object will permit patching is determined by the object's properties as described in Section 3.2.8.

It is recommended that the Server should not apply a patch to an object until all of the expected octets as specified by the *Length* parameter have been successfully received.

3.3.2.6.3 Handling Receipt of Unexpected Data

The Server knows the total number of octets to expect to receive from the Client as this is specified by the *Length* parameter.

In the event that the Server receives data via the Object Transfer Channel in excess of the expected number of octets, the Server shall close the Object Transfer Channel to prevent the Client from sending further data through the Object Transfer Channel. The Server may discard the data it has received.

3.3.2.7 Abort Procedure

When the *Abort* Op Code is written to the OACP and an error condition does not occur, the Server shall cease sending previously requested data to the Client through the Object Transfer Channel.

This allows the Client to abort the receipt of an object during an object transfer if it no longer requires to receive the data.

For this procedure, “Success” is defined to mean that the Server has accepted the request to abort an OACP Read procedure that is in progress and will cease sending the previously requested octets towards the Client through the Object Transfer Channel. The Server shall respond with the “Success” result code as soon as it has accepted the request.

However, if an error condition occurs, the “Success” result code shall not be sent and the appropriate error response shall be sent as defined in Table 3.11 and Table 3.20:

Error Condition	Error Response (See Section 3.3.2.8)
The *Abort* Op Code is not supported by the Server.	Op Code Not Supported
No OACP Read operation is in progress – there is nothing to abort.	Operation Failed
The Abort Op Code was not sent by the same Client who started the OACP Read operation.	Operation Failed
The requested procedure failed for a reason other than those enumerated above in this table.	Operation Failed

Table 3.20: *Requirements for Control Point Error Responses to the Abort Op Code*

In the event that more than one error condition listed in Table 3.20 occurs in the same transaction, the applicable Error Response which is nearest to the top of the table shall take priority and only this Error Response shall be sent.

Note that, since the intention of the OACP Abort procedure is to abort an OACP Read operation that is already in progress, the ATT Error response *Procedure Already in Progress* described in Section 3.3.2.8.1 is not sent if the OACP Read procedure is in progress when the Abort procedure is requested.

3.3.2.8 General Error Handling

Writing an Op Code to the OACP may result in an ATT Error response or a Control Point Error response as described below.

3.3.2.8.1 ATT Error

An ATT Write Request to the OACP characteristic may be rejected with an Attribute Protocol Error Response under certain conditions as defined in [1]. Reasons for sending an ATT Error Response include but are not limited to *Insufficient Encryption, Client Characteristic Configuration Descriptor Improperly Configured*, *Procedure Already in Progress*, or *Invalid Attribute Value Length.* See also Section 1.10 for Attribute Protocol Application Error codes defined by this service.

3.3.2.8.2 Control Point Error

If the ATT Write Request to the control point characteristic was successful but the requested control point procedure could not be performed or could not be completed successfully, an appropriate result code from Table 3.11 shall be indicated in the response as described above and in this section.

3.3.2.8.2.1 *Op Code Not Supported*

The *Op Code Not Supported* error response shall be indicated if the Op Code written to the control point is not supported by the Server. This includes Op Code values which are reserved for future use.

3.3.2.8.2.2 *Invalid Parameter*

The *Invalid Parameter* error response shall be indicated if the value of the parameter written to the control point did not meet the requirements of this service.

Note that a parameter of an incorrect size, including any parameter sent with an Op Code for which no parameter was actually required, should be trapped by an ATT Error response as described in Section 3.3.2.8.1 and in this case, it should not generate a control point error response.

3.3.2.8.2.3 *Insufficient Resources*

The *Insufficient Resources* error response shall be indicated if the number of octets requested via the value of the *Length* parameter exceeds the available memory or processing capabilities of the Server.

Note: The capabilities of the Server may be variable – for example, the available memory resources may vary over time.

3.3.2.8.2.4 *Invalid Object*

The *Invalid Object* error response shall be indicated in response to the *Calculate Checksum, Delete, Execute, Read*, *or Write* Op codes if the Current Object is an Invalid Object as defined in Section 1.8.

3.3.2.8.2.5 *Channel Unavailable*

The *Channel Unavailable* error response shall be indicated if a requested control point procedure cannot be performed because an Object Transfer Channel is not available for use. This may occur, for example, if the Client failed to open an Object Transfer Channel successfully before requesting an OACP *Read or Write* procedure.

3.3.2.8.2.6 *Unsupported Type*

The *Unsupported Type* error response shall be indicated if a requested control point procedure cannot be performed because the object type specified in the request is not supported by the Server.

3.3.2.8.2.7 *Procedure Not Permitted*

The *Procedure Not Permitted* error response shall be indicated if a requested control point procedure cannot be performed because it is not permitted according to the properties of the Current Object. This may occur, for example, if the Client attempts to use the *OACP Delete, Execute, Read, or Write* procedure when the properties of the Current Object do not permit the requested operation.

3.3.2.8.2.8 *Object Locked*

The *Object Locked* error response shall be indicated if a requested control point procedure cannot be performed because the Current Object has been temporarily locked by the Server. This may occur, for example, if the Server chooses to restrict access to the object while it is performing an internal operation or an operation initiated by another Client that is connected concurrently (see Section 1.12).

3.3.2.8.2.9 *Operation Failed*

The *Operation Failed* error response shall be indicated if a requested control point procedure failed for any other reason not otherwise enumerated.

3.3.2.8.3 Procedure Timeout

In the context of the OACP characteristic, a control point procedure is started when an ATT Write Request to the OACP characteristic is successful (i.e., when the Server sends the ATT Write Response).

The Server may then use an implementation-specific timeout. If the timeout expires before the requested control point procedure has been completed, the Server may abort the service procedure and indicate the *Operation Failed* error response described in Section 3.3.2.8.2.9.

A control point procedure is not considered started and not queued in the Server when a Write Request to the OACP characteristic results in an ATT Error response as described in Section 3.3.2.8.1.

3.4 Object List Control Point (OLCP)

The structure of the OLCP characteristic is defined below:

LSO MSO

	Op Code	**Parameter**
	(See also Table 3.22.)	
Octet Order	N/A	LSO...MSO
Data type	UINT8	Variable
Size	1 octet	0 to 6 octets
Units	None	None

Table 3.21: *Structure of the Object List Control Point*

3.4.1 Object List Control Point Procedure Requirements

Table 3.22 shows the requirements for the OLCP characteristic:

Op Code Value	**Procedure**	**Require-ment**	**Parameter**	**Applicable Response Value**	**Response Parameter**
0x00	Reserved for Future Use				
0x01	First	M	None	Success, Operation Failed, Too Many Objects, No Object	None
0x02	Last	M	None	Success, Operation Failed, Too Many Objects, No Object	None

Op Code Value	Procedure	Require-ment	Parameter	Applicable Response Value	Response Parameter
0x03	Previous	M	None	Success, Out Of Bounds, Operation Failed, Too Many Objects, No Object	None
0x04	Next	M	None	Success, Out Of Bounds, Operation Failed, Too Many Objects, No Object	None
0x05	Go To	O	Object ID (UINT48)	Success, Invalid Parameter, Object ID Not Found, Op Code Not Supported, Operation Failed, Too Many Objects, No Object	None
0x06	Order	O	List Sort Order (UINT8) (see Table 3.23)	Success, Invalid Parameter, Op Code Not Supported, Operation Failed, Too Many Objects, No Object	None
0x07	Request Number of Objects	O	None	Success, Op Code Not Supported, Operation Failed, Too Many Objects, No Object	Total Number of Objects (UINT32)
0x08	Clear Marking	O	None	Success, Op Code Not Supported, Operation Failed, Too Many Objects, No Object	None
0x09-0x6F	Reserved for Future Use				
0x70	Response Code	M	OLCP Response value (see Table 3.25)	N/A	N/A

Op Code Value	Procedure	Require-ment	Parameter	Applicable Response Value	Response Parameter
0x71-0xFF	Reserved for Future Use				

Table 3.22: *Object List Control Point Procedure Requirements*

The *List Sort Order* Parameter (UINT8) has the following defined values:

Parameter Value	Description
0x00	Reserved for future use
0x01	Order the list by object name, ascending (Note 1)
0x02	Order the list by object type, ascending
0x03	Order the list by object current size, ascending
0x04	Order the list by object first-created timestamp, ascending
0x05	Order the list by object last-modified timestamp, ascending
0x06-0x10	Reserved for future use
0x11	Order the list by object name, descending (Note 1)
0x12	Order the list by object type, descending
0x13	Order the list by object current size, descending
0x14	Order the list by object first-created timestamp, descending
0x15	Order the list by object last-modified timestamp, descending
0x16-0xFF	Reserved for future use

Table 3.23: *List Sort Order - Enumeration of Parameter Values*

Note 1: For consistency between implementations, the sort algorithms used for ordering strings in ascending or descending alphabetical order should follow the guidance in ISO/IEC 14651 [5].

The response to the Object List Control Point is of variable length and is formatted as follows:

LSO MSO

	OLCP Response Value		
	Request Op Code	Result Code	Response Parameter (if present)
		(See also Table 3.25.)	
Octet Order	N/A	N/A	LSO…MSO
Data type	UINT8	UINT8	UINT32
Size	1 octet	1 octet	0 or 4 octets

Table 3.24: *Format of the OLCP Response Value*

The Result Codes and Response Parameters associated with Op Code 0x70 are defined as:

Result Code	Definition	Response Parameter	Description
0x00	Reserved For Future Use		
0x01	Success	None except for Op Code 0x07 (Request Number of Objects) (UINT32)	Response for successful operation.
0x02	Op Code Not Supported	None	Response if unsupported Op Code is received.
0x03	Invalid Parameter	None	Response if Parameter received does not meet the requirements of the service.
0x04	Operation Failed	None	Response if the requested procedure failed for a reason other than those enumerated below.
0x05	Out Of Bounds	None	Response if the requested procedure attempted to select an object beyond the first object or beyond the last object in the current list.
0x06	Too Many Objects	None	Response if the requested procedure failed due to too many objects in the current list.
0x07	No Object	None	Response if the requested procedure failed due to there being zero objects in the current list.

v10

Result Code	Definition	Response Parameter	Description
0x08	Object ID Not Found	None	Response if the requested procedure failed due to there being no object with the requested Object ID.
0x09-0xFF	Reserved For Future Use		

Table 3.25: *List of OLCP Result Codes and Response Parameters*

3.4.2 Object List Control Point Characteristic Behavior

The Object List Control Point (OLCP) is used by a Client to control certain behaviors of the Server. The role of the OLCP is to provide a mechanism for the Client to find the desired object and to designate it as the Current Object. The OLCP procedures shall not modify the contents of an object or its metadata.

The procedures are triggered by writing a value that includes an Op Code specifying the operation and this may be followed by a Parameter that is valid within the context of that Op Code (see Table 3.22).

Designating an object as the Current Object by using the procedures of the OLCP shall cause the metadata of that object to be exposed via the Object Metadata characteristics as described in Section 3.1.

Each OLCP procedure is described below. Unless otherwise stated, when any OLCP procedure fails, the object selected as the Current Object shall remain the same as it was prior to the procedure being attempted.

For each of these procedures, if the procedure is successfully completed, the Server shall indicate the OLCP characteristic with the Success result code contained in the response. If the procedure is not successfully completed, an error response shall be returned as described in Section 3.4.2.9.

3.4.2.1 First Procedure

When the *First* Op Code is written to the OLCP, the Server shall make the first object in the list become the Current Object.

3.4.2.2 Last Procedure

When the *Last* Op Code is written to the OLCP, the Server shall make the last object in the list become the Current Object.

3.4.2.3 Previous Procedure

When the *Previous* Op Code is written to the OLCP, the Server shall make the object immediately preceding the Current Object in the list become the Current Object.

If this procedure is used when the Current Object is already the first in the list, it shall remain unchanged.

The “Out Of Bounds” response value described in Section 3.4.2.9.2.4 is applicable if the Current Object was the first object when the *Previous* Op Code was written.

3.4.2.4 Next Procedure

When the *Next* Op Code is written to the OLCP, the Server shall make the object immediately after the Current Object in the list become the Current Object.

If this procedure is used when the Current Object is already the last in the list, it shall remain unchanged.

The “Out Of Bounds” response value described in Section 3.4.2.9.2.4 is applicable if the Current Object was the last object when the *Next* Op Code was written.

3.4.2.5 Go To Procedure

This procedure enables a Client to select an object directly by specifying its Object ID. The *Go To* Op Code requires a parameter, *Object ID*.

For the purposes of this procedure, any filter conditions exposed via the Object List Filter characteristics, if supported, are ignored; in other words, if the object identified by the Object ID is being excluded from the list of objects by the filter conditions, this shall not prevent the Go To procedure from succeeding.

If the response to the *Go To* Op Code is “Success”, the value of the Object List Filter characteristics, if supported, shall be updated by the Server to ensure that the filter type is set to “No Filter” and the object specified in the *Object ID* parameter shall be designated as the Current Object.

When the *Go To* Op Code is written to the OLCP and the procedure is supported by the Server, the Server shall make the object with the specified *Object ID* become the Current Object.

However, if no object is found on the Server with the specified Object ID, the *Object ID Not Found* error response shall be indicated.

3.4.2.6 Order Procedure

When the *Order* Op Code is written to the OLCP and re-ordering of the list is supported by the Server, the Server shall arrange the list of objects in order according to the value of the *List Sort Order* parameter. The assigned values for *List Sort Order* are enumerated in Table 3.23 above.

Note that only a different sort order is presented and no change to any object or metadata is made.

The object that was the Current Object prior to the Order procedure shall remain the Current Object after the completion of this procedure.

Any First, Last, Previous, or Next procedures used after the list has been re-ordered shall henceforth work in accordance with the new list order.

3.4.2.7 Request Number of Objects Procedure

When the *Request Number of Objects* Op Code is written to the OLCP and the reporting of the number of objects is supported, the Server shall report the total number of objects found in the list of objects.

If the Object List Filter characteristics described in Section 3.5 are supported, only objects included after the application of the filter conditions shall be included in the count of the number of objects. Note that a Client can use the *No Filter* filter type described in Section 3.5.1.1 when it is desired to obtain the unfiltered total.

If the procedure is successful, the Success response shall include a parameter containing the count of the total number of objects as described in Table 3.25.

3.4.2.8 Clear Marking Procedure

Marked objects are identified by the Mark property bit defined in Section 3.2.8 having the value 1 (True). When the *Clear Marking* Op Code is written to the OLCP and the clearing of marking is supported, the Server shall update the marking of objects so that all objects in the list of objects become unmarked in respect of the Client that sent the *Clear Marking* Op Code.

If the Object List Filter characteristics described in Section 3.5 are supported, only objects that remain in the list of objects after the application of the filter conditions shall be included in the Clear Marking procedure.

As a result, the Mark property bit for all objects included in this procedure shall be set to 0 (False).

The Clear Marking procedure enables a Client to clear the marks against objects that it has processed. Refer to Section 3.5.1.11 for further information.

3.4.2.9 General Error Handling

Writing an Op Code to the OLCP may result in an ATT Error response or a Control Point Error response as described below.

The applicability of Error Responses depends on the OLCP procedure and is specified in Table 3.22. Since it is possible for more than one error condition to occur in the same transaction, the applicable Error Response which is nearest to the top of Table 3.26 shall take priority and only this Error Response shall be sent:

Priority	Error Response	Requirement Section
1	ATT Error	3.4.2.9.1
2	Op Code Not Supported	3.4.2.9.2.1
3	No Object	3.4.2.9.2.6
4	Too Many Objects	3.4.2.9.2.5
5	Out Of Bounds	3.4.2.9.2.4
6	Object ID Not Found	3.4.2.9.2.7
7	Invalid Parameter	3.4.2.9.2.2
8	Operation Failed	3.4.2.9.2.3

Table 3.26: *Order of Priority of OLCP Error Responses*

3.4.2.9.1 ATT Error

An ATT Write Request to the OLCP characteristic may be rejected with an Attribute Protocol Error Response under certain conditions as defined in [1]. Reasons for sending an ATT Error Response include but are not limited to *Insufficient Authentication, Client Characteristic Configuration Descriptor Improperly Configured*, or *Invalid Attribute Value Length.* See also Section 1.10 for Attribute Protocol Application Error codes defined by this service.

3.4.2.9.2 Control Point Error

If the ATT Write Request to the control point characteristic was successful but the requested control point procedure could not be performed or could not be completed successfully, an appropriate result code from Table 3.25 shall be indicated in the response as described in the following sub-sections.

3.4.2.9.2.1 *Op Code Not Supported*

The *Op Code Not Supported* error response shall be indicated if the Op Code written to the control point is not supported by the Server. This includes Op Code values which are reserved for future use.

3.4.2.9.2.2 *Invalid Parameter*

The *Invalid Parameter* error response shall be indicated if the value of the parameter written to the control point did not meet the requirements of this service.

Note that a parameter of an incorrect size, including any parameter sent with an Op Code for which no parameter was actually required, should be trapped by an ATT Error response as described in Section 3.3.2.8.1 and in this case, it should not generate a control point error response.

3.4.2.9.2.3 *Operation Failed*

The *Operation Failed* error response shall be indicated if a requested control point procedure failed for any other reason not otherwise enumerated.

3.4.2.9.2.4 *Out Of Bounds*

The *Out Of Bounds* error response shall be indicated if it was attempted to select an object beyond the first object in the current list or beyond the last object in the current list.

If the Object List Filter characteristics described in Section 3.5 are supported, the current list in this context means the objects that remain in the list of objects after the application of the filter conditions.

3.4.2.9.2.5 *Too Many Objects*

The *Too Many Objects* error response shall be indicated if the control point procedure could not be performed successfully because the Server had too many objects to handle.

3.4.2.9.2.6 *No Object*

The No Object error response shall be indicated if the requested control point procedure could not be performed because there was no object in the list. This may include a list which has been rendered empty as the result of the application of the filter conditions described in Section 3.5 below.

3.4.2.9.2.7 *Object ID Not Found*

The *Object ID Not Found* error response shall be indicated if the Object ID specified in the *Object ID* parameter of the Op Code does not match the Object ID of any object on the Server.

3.4.2.9.3 Procedure Timeout

In the context of the OLCP characteristic, a control point procedure is started when an ATT Write Request to the OLCP characteristic is successful (i.e., when the Server sends the ATT Write Response).

The Server may then use an implementation-specific timeout. If the timeout expires before the requested control point procedure has been completed, the Server may abort the procedure and indicate the *Operation Failed* error response described in Section 3.4.2.9.2.3.

A control point procedure is not considered started and not queued in the Server when a Write Request to the OLCP characteristic results in an ATT Error response as described in Section 3.4.2.9.1.

3.5 Object List Filter Characteristic

If Object List Filter characteristics are exposed, the filter conditions shall determine which objects are included in or excluded from the list of objects. The Object List Filter characteristics can therefore modify the behavior of the Object List Control Point. The Object List Filter characteristics thus enable criteria to be set that determine which objects are exposed to the Client.

If the Server supports the Object List Filter characteristic, it shall support three instances of this characteristic as described in Section 3.5.1.

Each Object List Filter characteristic is a variable length characteristic that contains up to two fields: a Filter field and a Parameter field.

The structure of each Object List Filter characteristic is defined in Table 3.27:

LSO MSO

	Filter	**Parameter**
	(See also Table 3.30.)	
Octet Order	N/A	LSO … MSO
Data type	UINT8	(Variable - defined per Filter Value)
Size	1 octet	variable (defined per Filter Value)

Table 3.27: *Structure of the Object List Filter Characteristic*

The format of the Parameter depends on the Filter value.

The filter conditions that can be set are enumerated in Table 3.28. If the Server supports the Object List Filter characteristic, it shall support all of the following filter conditions. In case the Server does not support the relevant object metadata, see the relevant sub-section in Section 3.5.1 for the required behavior.

Filter Value	**Filter Description**	**Parameter Description**
0x00	No Filter (everything passes)	N/A
0x01	Name Starts With	String (UTF-8)
0x02	Name Ends With	String (UTF-8)
0x03	Name Contains	String (UTF-8)

Filter Value	Filter Description	Parameter Description	
0x04	Name is Exactly	String (UTF-8)	
0x05	Object Type (UUID)	UUID (gatt_uuid)	
0x06	Created between (inclusive): timestamp1 <= t <= timestamp2	LSO	MSO
		timestamp1 (same format as the Date Time characteristic in [3])	timestamp2 (same format as the Date Time characteristic in [3])
0x07	Modified between (inclusive): timestamp1 <= t <= timestamp2	LSO	MSO
		timestamp1 (same format as the Date Time characteristic in [3])	timestamp2 (same format as the Date Time characteristic in [3])
0x08	Current Size between (inclusive): size1 <= s <= size2	LSO	MSO
		size1 (UINT32)	size2 (UINT32)
0x09	Allocated Size between (inclusive): size1 <= s <= size2	LSO	MSO
		size1 (UINT32)	size2 (UINT32)
0x0A	Marked Objects	N/A	
0x0B-0xFF	Reserved for future use	N/A	

Table 3.28: *Filter Conditions used in the Object List Filter Characteristic Value*

3.5.1 Object List Filter Characteristic Behavior

The type of filter to be applied is identified by the Filter Value field. The filters are described below.

The default value for each Object List Filter characteristic shall be *No Filter.* Upon first connection and upon every reconnection, the value of each Object List Filter characteristic shall be reset to 0x00 (No Filter); hence, there is no persistence of the value of this characteristic. When the Object List Filter characteristics are supported, the Server shall allow new values to be written to the Filter and Parameter fields.

The filter conditions exposed via the three instances of the Object List Filter characteristic shall be combined by using logical “AND” operations.

It is possible that the Current Object will not satisfy the newly applied filter conditions and requirements covering this situation are specified in Section 3.5.2.

If the Server supports the Concurrency Feature, refer also to Section 1.12 for additional requirements.

3.5.1.1 No Filter

When the filter type is *No Filter*, it shall have no filtering effect. Therefore, when the filter type of all supported Object List Filter characteristics is *No Filter*, all the objects available on the Server shall be included in the list.

3.5.1.2 Name Starts With (UTF-8 string)

When the filter type is *Name Starts With*, only those objects whose name starts with the string provided in the Parameter field shall be included in the list. The comparison shall be case-sensitive.

3.5.1.3 Name Ends With (UTF-8 string)

When the filter type is *Name Ends With*, only those objects whose name ends with the string provided in the Parameter field shall be included in the list. The comparison shall be case-sensitive.

3.5.1.4 Name Contains (UTF-8 string)

When the filter type is *Name Contains*, only those objects whose name contains the string provided in the Parameter field shall be included in the list. The comparison shall be case-sensitive.

3.5.1.5 Name is Exactly (UTF-8 string)

When the filter type is *Name is Exactly*, only those objects whose name exactly matches the string provided in the Parameter field shall be included in the list. The comparison shall be case-sensitive.

3.5.1.6 Object Type (UUID)

When the filter type is *Object Type*, only those objects whose object type matches the UUID provided in the Parameter field shall be included in the list.

v10

3.5.1.7 Created Between (timestamp1, timestamp2)

When the filter type is *Created Between*, only those objects whose First-Created timestamp value falls between the two values provided in the Parameter field, inclusive, shall be included in the list.

However, objects whose Object First-Created metadata are set to a special value indicating that the date and/or time are not valid as defined in Section 3.3.2.1 and objects whose Object First-Created metadata does not exist shall not be excluded from the list by the application of this filter type.

Timestamp1 shall represent a time that is earlier than or equal to timestamp2. If the Client attempts to write a value to an Object List Filter characteristic in which timestamp1 represents a time that is later than timestamp2, the Server shall reject such a write request by sending the applicable Attribute Protocol Application Error code “Write Request Rejected” as defined in Section 1.10.

3.5.1.8 Modified Between (timestamp1, timestamp2)

When the filter type is *Modified Between*, only those objects whose Last-Modified timestamp value falls between the two values provided in the Parameter field, inclusive, shall be included in the list.

However, objects whose Object Last-Modified metadata are set to a special value indicating that the date and/or time are not valid as defined in Section 3.3.2.1 and objects whose Object Last-Modified metadata does not exist shall not be excluded from the list by the application of this filter type.

Timestamp1 shall represent a time that is earlier than or equal to timestamp2. If the Client attempts to write a value to an Object List Filter characteristic in which timestamp1 represents a time that is later than

timestamp2, the Server shall reject such a write request by sending the applicable Attribute Protocol Application Error code “Write Request Rejected” as defined in Section 1.10.

3.5.1.9 Current Size Between (size1, size2)

When the filter type is *Current Size Between*, only those objects whose Current Size value falls between the two values provided in the Parameter field, inclusive, shall be included in the list.

The value of size1 shall be less than or equal to the value of size2. If the Client attempts to write a value to an Object List Filter characteristic in which the value of size1 is greater than size2, the Server shall reject such a write request by sending the applicable Attribute Protocol Application Error code “Write Request Rejected” as defined in Section 1.10.

3.5.1.10 Allocated Size Between (size1, size2)

When the filter type is *Allocated Size Between*, only those objects whose Allocated Size value falls between the two values provided in the Parameter field, inclusive, shall be included in the list.

The value of size1 shall be less than or equal to the value of size2. If the Client attempts to write a value to an Object List Filter characteristic in which the value of size1 is greater than size2, the Server shall reject such a write request by sending the applicable Attribute Protocol Application Error code “Write Request Rejected” as defined in Section 1.10.

3.5.1.11 Marked Objects

When the filter type is *Marked Objects*, only those objects which are marked shall be included in the list.

This allows the Server to mark objects for a Client. The fact that an object has been marked can be identified by reading the associated bit of the Object Properties characteristic as described in Section 3.2.8 and the use of the Marked Objects filter type described here allows a Client to find efficiently all the objects that the Server has marked for it.

If the Server has bonded Clients, it shall manage the marking and unmarking of objects on a per bond basis.

The reasons for marking objects and the usage of this mechanism by applications will depend on the profile and may be defined by a higher-level specification. For example, the Server may have an application that enables the user to bookmark certain objects which form a subset of the objects available on the Server; the Client is then able to identify which objects are bookmarked.

3.5.2 Excluding the Current Object

If a new value written to an Object List Filter characteristic updates the filter conditions in such a way that it causes the Current Object to be excluded from the list of objects, this action shall make the Current Object invalid.

The Current Object shall remain invalid until the filter conditions are changed again such that the Current Object is not excluded any longer or an object is selected from the filtered list to become the new Current Object.

The handling of requests for OACP operations to be performed on Invalid Objects is described in Section 3.3.2.8.2.4.

3.6 Object Changed Characteristic

The Object Changed characteristic enables a Client to receive an indication if the contents and/or metadata of one or more objects are changed while the Client is connected without the change being made by an action of the Client itself. This allows the Client to know that the object database has changed while it is connected.

A Flags field is included in the characteristic value. This is used to show the nature of the change.

The structure of the Object Changed characteristic is defined in Table 3.29:

LSO MSO

	Flags	Object ID
Octet Order	N/A	LSO … MSO
Format type	8bit	UINT48
Size	1 octet	6 octets
Units	Unitless	Unitless

Table 3.29: *Object Changed Characteristic - Structure*

3.6.1 Object Changed Characteristic Behavior

An object and/or its metadata can become changed either when a value is changed by another Client or a change is made locally at the Server itself (e.g., via the Server UI).

When the Object Changed characteristic is configured for indications via the Client Characteristic Configuration descriptor, the Server shall send indications of the Object Changed characteristic, subject to the restrictions shown in Table 3.30. If multiple objects have changes, multiple indications may be sent.

An indication shall be sent only to Clients that are connected to the Server at the time that the change occurs. The Object Changed characteristic shall be indicated to the connected Clients when any object available on the Server changes, with the exception of the special Directory Listing Object described in Section 4.1.

Table 3.30 summarizes the required behavior depending on the source of the change:

Change Occurred at Server (Bit 0 of Flags = 0)	Change Made by Another Client (Bit 0 of Flags = 1)
Indicate Object Changed to all Clients.	Indicate Object Changed to Clients other than the Client that made the change.

Table 3.30: *Object Changed Characteristic – Behavior*

3.6.1.1 Flags Field

The Flags field shall be included in the Object Changed characteristic.

Reserved for Future Use (RFU) bits in the Flags fields shall be set to 0.

The bits of the Flags field and their function are shown in Table 3.31.

Bit	Definition
0	Source of Change 0: Server 1: Client
1	Change occurred to the object contents 0: False 1: True
2	Change occurred to the object metadata 0: False 1: True
3	Object Creation 0: False 1: True
4	Object Deletion 0: False 1: True
5-7	Reserved for Future Use

Table 3.31: *Object Changed Characteristic – Flags*

v10

Bit 0 (Source of Change) shall be set according to which device made the change. This allows the Client to inform the user about the source of change.

Bits 1 and 2 shall be set according to which aspects of the object were changed. If both the object's contents and its metadata have changed, both these bits shall be set. However, if the change consists only of the creation or deletion of an object, this shall be signaled via bit 3 or 4, as applicable, and bits 1 and 2 shall not be set in this case.

Bit 3 (Object Creation) shall be set if the change consists of the creation of a new object. This allows the Client to know that a new object has become available on the Server.

Bit 4 (Object Deletion) shall be set if the change consists of an object having been deleted from the Server. This allows the Client to know that an object that was previously available will no longer be found on the Server.

3.6.1.2 Object ID Field

The Object ID field shall be included in the Object Changed characteristic.

The value of the Object ID field shall be the Object ID of the object to which the relevant change applies. The value shall not be equal to 0.

3.7 Procedure Following Disconnection

This section specifies the action that the Server shall take following the termination of a connection.

A connection may be terminated intentionally or following a break-down of the connection, detected by the expiry of the Link Layer connection supervision timer. Whenever a connection is terminated, whatever the cause, the Server shall take the following action.

Following termination of a connection, the Server is not required to cache any Object List Filter characteristic value that was written by the Client nor which object was selected by the Client as the Current Object during that connection, (i.e., it is not required that such values persist until the Client reconnects).

If the OACP Create procedure was used during the connection session that has just been terminated, the Server shall check the value of the name metadata of any object created during that connection. If the name metadata of any such object still consists of a zero-length string, the object shall be deleted; this deletion shall occur even if the object possesses properties that normally do not permit deletion (i.e., even if the Delete property of the object is set to False). This action is required to ensure that any newly created object that does not meet the minimum requirements to be an object at the time a connection ends is deleted from the Server.

If the Server supports the Concurrency Feature described in Section 1.12 and an object had been locked on behalf of the Client that has now become disconnected, the Server shall remove the lock.

If an object transfer is in progress when a disconnection occurs, refer also to the Object Transfer Timeout Procedure described in Section 3.8.

3.8 Object Transfer Timeout Procedure

3.8.1 Object Transfer Timeout – Object Being Written

During execution of the OACP Write procedure described in Section 3.3.2.6, the Server is informed of the total number of octets to expect from the Client through the Object Transfer Channel - this is specified by the *Length* parameter of the OACP Write procedure.

If the Server successfully receives the expected number of octets, the object transfer shall be considered completed. The integrity of the data transferred can be verified as described in Section 3.3.2.3 or by other means.

However, if fewer than the expected number of octets have been received and the receipt of data from the Client through the Object Transfer Channel has ceased:

(a) due to link loss; or
(b) because the Object Transfer Channel has been closed unexpectedly by the Client; or
(c) when no data has been received for a period longer than 30 seconds,

then the object transfer shall be deemed to have been aborted with the transfer only partially completed and the Server shall close the Object Transfer Channel, if open.

The Server should store the incomplete object. If it is stored, the object transfer can be resumed later as described in Section 3.2.4.1.1.

If the Server supports the Concurrency Feature described in Section 1.12 and the object was locked, the Server should remove the lock; this may occur after an implementation-specific delay.

3.8.2 Object Transfer Timeout – Object Being Read

During execution of the OACP Read procedure described in Section 3.3.2.5, the Server is informed of the total number of octets to send to the Client through the Object Transfer Channel - this is specified by the *Length* parameter of the OACP Read procedure.

If the Server successfully sends the expected number of octets, the object transfer shall be considered completed. The integrity of the data transferred can be verified as described in Section 3.3.2.3 or by other means.

Reasons that the Server may be prevented from sending data to the Client through an Object Transfer Channel include, for example, link loss or, when an LE L2CAP Connection Oriented Channel is used, a failure by the Client to allocate sufficient credit for the LE Credit Based Flow Control.

If fewer than the expected number of octets has been sent and the sending of data from the Server through the Object Transfer Channel has been prevented:

(a) due to link loss; or
(b) because the Object Transfer Channel has been closed unexpectedly by the Client; or
(c) when the sending of data has been prevented for a period longer than 30 seconds,

then the object transfer shall be deemed to have ended unexpectedly, with the transfer only partially completed and the Server shall close the Object Transfer Channel, if open.

v10

4 Special Objects

4.1 Directory Listing Object

The Directory Listing Object is intended for Servers that support a large number of objects to facilitate the efficient provision of a complete directory listing to the Client.

Only Servers that are capable of storing more than one object may support the Directory Listing Object.

The Server may expose no more than one Directory Listing Object and, if exposed, it shall be assigned the Object ID of 0x000000000000.

The Directory Listing Object shall contain a complete listing of the objects available on the Server. The contents and format of the Directory Listing Object shall conform to the definition of the <<Directory Listing>> Object Type given in Section 4.1.1.

In the event that the Object Server currently contains no other object, the Directory Listing Object shall contain a single Object Record, listing only itself.

If the Server exposes the Directory Listing Object, it shall maintain the contents of this object up to date when any change occurs to the object database.

This enables the Client to obtain a current listing of the available objects by reading the Directory Listing Object, using the features of this service to select and read the object contents.

The object metadata associated with the Directory Listing Object shall conform to the following:

- The Object Type shall be <<Directory Listing>> as defined in [3].
- The Object ID for this object shall have the value 0x000000000000.
- The Object Size shall be exposed.
- The Object Properties shall only permit reading of this object (i.e., the object properties shall not permit a client to modify the object contents nor to delete the object).
- The Object Name should reflect the nature of the object. It is recommended to use “Directory” or an equivalent name, in the language supported by the implementation.

v10

4.1.1 Directory Listing Object Format

The Directory Listing Object shall consist of one or more Object Records where each “Object Record” has the structure defined in Table 4.1. Each Object Record reports the object name and other relevant metadata for an individual object. This structure is repeated within the Directory Listing Object until all objects have been listed, concatenating the Object Records together to form the object contents.

The Object Record structure as shown in Table 4.1 is of variable length. Fields marked “M” (mandatory) in Table 4.1 shall be included in every Object Record and fields marked “O” (optional) may be included in the object record. The presence of optional fields may differ from one Object Record to another within the Directory Listing Object contents. The presence of optional fields shall be specified for each Object Record by the bits of the Flags field as described in Section 4.1.1.5.

The Object Record ordering follows the Little Endian format, with the LSO being the beginning of the first Object Record.

	LSO											MSO
	Length of Object Record (octets)	**Object ID**	**Object Name Length (octets)**	**Object Name**	**Flags**	**Object Type**	**Current Size**	**Allocated Size**	**Object First-Created**	**Object Last-Modified**	**Object Properties**	**Extension (RFU)**
Octet Order	LSO...MSO	LSO...MSO	N/A	LSO...MSO	N/A	LSO...MSO	LSO...MSO	LSO...MSO	See Section 4.1.1.9	See Section 4.1.1.10	LSO...MSO	See Section 4.1.1.12
Data type	UINT16	UINT48	UINT8	UTF-8	8bit	gatt_uuid	UINT32	UINT32	See Section 4.1.1.9	See Section 4.1.1.10	32bit	See Section 4.1.1.12
Size	2 octets	6 octets	1 octet	variable : 1 to 120 octets	1 octet	2 or 16 octets	4 octets	4 octets	7 octets	7 octets	4 octets	0 octets
Field status	M	M	M	M	M	M	O	O	O	O	O	RFU

Table 4.1: *Directory Listing Object – Object Record Structure*

It is recommended that the objects be ordered in ascending alphabetical order by object name and, for consistency, the alphabetization should follow the guidance in ISO/IEC 14651 [5].

The Directory Listing Object provides a flat listing of objects; there is no path or folder / sub-folder structure information specified.

The fields of an Object Record are defined in the following sub-sections:

4.1.1.1 Length of Object Record

The Object Record structure is of variable length. Each Object Record shall begin with a “Length of Object Record” field that specifies the total size of the current Object Record in octets. The total value reported includes the 2 octets used by the “Length of Object Record” field itself. The length value enables a client to locate the beginning of the next Object Record.

4.1.1.2 Object ID

The format of the Object ID field shall be the same as the format of the value of the Object ID characteristic that is defined in Section 3.2.7.

4.1.1.3 Object Name Length

The Object Name Length field shall specify the size in octets of the Object Name field that immediately follows it.

4.1.1.4 Object Name

The format of the Object Name field shall be the same as the format of the value of the Object Name characteristic that is defined in Section 3.2.2.

4.1.1.5 Flags

The Flags field shall be included in each Object Record.

Any RFU (Reserved for Future Use) bits in the Flags fields shall be set to 0.

The bits of the Flags field and their function are shown in Table 4.2:

Bit Number	Definition
0	Object Type UUID Size 0: 16bit 1: 128bit
1	Current Size Present 0: False 1: True
2	Allocated Size Present 0: False 1: True
3	Object First-Created Present 0: False 1: True
4	Object Last-Modified Present 0: False 1: True
5	Object Properties Present 0: False 1: True
6	RFU
7	Extended Flags Present 0: False 1: True

Table 4.2: Directory Listing Object – Object Record - Flags

The value of the Extended Flags Present field will always be 0 (False) in this version of the service. This flag has been included to support forwards compatibility. If Extended Flags are defined in a future version of this service, they may be included in the Extension field described in Section 4.1.1.12.

4.1.1.6 Object Type

The format of the Object Type field shall be the same as the format of the value of the Object Type characteristic that is defined in Section 3.2.3.

The length of the Object Type field (being either 16bit or 128bit) is specified by Bit 0 of the preceding Flags field.

4.1.1.7 Current Size

The format of the Current Size field shall be the same as the format of the Current Size field of the Object Size characteristic that is defined in Section 3.2.4.

4.1.1.8 Allocated Size

The format of the Allocated Size field shall be the same as the format of the Allocated Size field of the Object Size characteristic that is defined in Section 3.2.4.

4.1.1.9 Object First-Created

The format of the Object First-Created field shall be the same as the format of the value of the Object First-Created characteristic that is defined in Section 3.2.5.

4.1.1.10 Object Last-Modified

The format of the Object Last-Modified field shall be the same as the format of the value of the Object Last-Modified characteristic that is defined in Section 3.2.6.

4.1.1.11 Object Properties

The format of the Object Properties field shall be the same as the format of the value of the Object Properties characteristic that is defined in Section 3.2.8.

4.1.1.12 Extension

In this version of the service, the Extension field is never used and shall be of zero length. However, it is reserved for use in a future version of this specification. The position of the Extension field has been included in Table 4.1 to define that all the other fields of an Object Record shall be sent before the Extension field, if any, so that a Collector is able to identify the presence of any RFU octets in the structure and ignore them, using the value provided in the Length of Object Record field to identify the end of the present Object Record and the start of the next.

5 SDP Interoperability

If this service is exposed over BR/EDR then it shall have the following SDP record.

Item	Definition	Type	Value	Status
Service Class ID List				M
Service Class #0		UUID	«Object Transfer Service»	M
Protocol Descriptor List				M
Protocol #0		UUID	«L2CAP»	M
Parameter #0 for Protocol #0	PSM	UINT16	PSM = ATT	M
Parameter #1 for Protocol #0	PSM	UINT16	PSM = PSM_OTS	M
Protocol #1		UUID	ATT	M
Parameter #0 for Protocol #1	GATT Start Handle	UINT16	First handle of this service in the GATT database	M
Parameter #1 for Protocol #1	GATT End Handle	UINT16	Last handle of this service in the GATT database	M
BrowseGroupList			PublicBrowseRoot*	M

Table 5.1 *SDP Record*

* PublicBrowseRoot shall be present; however, other browse UUIDs may also be included in the list.

v10

6 Acronyms and Abbreviations

Abbreviation or Acronym	Meaning
ATT	Attribute Protocol
GATT	Generic Attribute Profile
LE	Low Energy
LUID	Locally Unique Identifier
PSM	Protocol Service Multiplexer
RFU	Reserved for Future Use
UUID	Universally Unique Identifier

Table 6.1: *Acronyms and Abbreviations*

7 References

[1] Bluetooth Core Specification; v4.0 or later version.

[2] Bluetooth Core Specification; v4.1 or later version.

[3] Characteristic and Descriptor descriptions are accessible via the Bluetooth SIG Assigned Numbers.

[4] ISO/IEC 3309: Information technology - Telecommunications and information exchange between systems - High-level data link control (HDLC) procedures; 1993 or later version.

[5] ISO/IEC 14651: Information technology - International string ordering and comparison - Method for comparing character strings and description of the common template tailorable ordering; 2011 or later version.

v10