Object Transfer Profile

Bluetooth® Profile Specification

- **Date** 2015-Nov-17
- **Revision** v10
- **Group Prepared By** SF WG

Abstract:

This Bluetooth profile defines fundamental requirements to enable an Object Client to create and delete objects and to execute a variety of actions using the currently selected object such as reading object data from or writing object data to an Object Server that exposes the Object Transfer Service. This profile is designed to be referenced by a higher layer specification to enable a variety of object transfer use cases.

Revision History

Revision Number	Date	Comments
v10	2015-11-17	Adopted by the Bluetooth SIG BoD

Contributors

Name	Company
Robert D. Hughes	Intel Corporation
Laurence Richardson	Qualcomm Technologies, Inc.
Guillaume Schatz	Polar Electro Oy
Leif-Alexandre Aschehoug	Nordic Semiconductor ASA
Charlie Lee	ISSC Technologies Corporation
Navin Kochar	Intel 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.

v10

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.

Contents

v10

v10

1 Introduction

The Object Transfer Profile defines fundamental requirements to enable an Object Client to create and delete objects and to execute a variety of actions using the currently selected object such as reading object data from or writing object data to an Object Server that exposes the Object Transfer Service [1]. This profile is designed to be referenced by a higher layer specification to enable a variety of object transfer use cases.

1.1 Profile Dependencies

This profile requires the Generic Attribute Profile (GATT).

1.2 Conformance

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

1.3 Bluetooth Specification Release Compatibility

This specification is compatible with the Bluetooth Core Specification v4.0 [2] or later when data is transferred using a BR/EDR transport and the Bluetooth Core Specification v4.1 [3] or later when data is transferred using an LE transport.

v10

2 Configuration

2.1 Roles

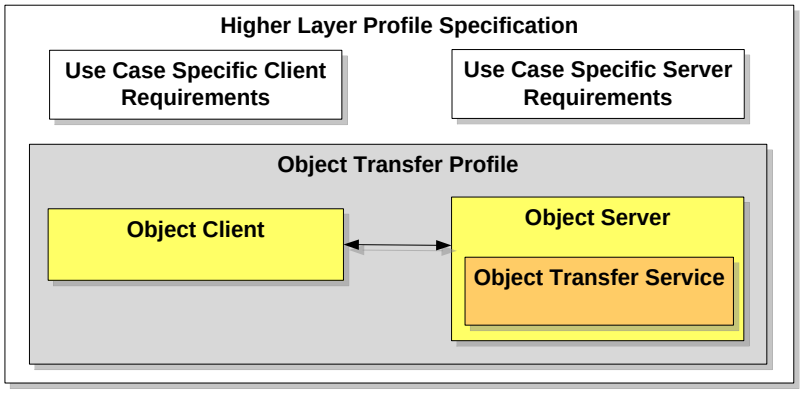
The profile defines two roles: Object Server and Object Client.

The Object Server is the device that interacts with an Object Client for the reading or writing of object data and exposes object metadata. The Object Client is the device that initiates reading, writing or other object action from the Object Server.

- The Object Server shall be a GATT Server
- The Object Client shall be a GATT Client

2.2 Role/Service Relationships

The following diagram shows the relationships between service and profile roles.



Note: Profile roles (Object Client, Object Server) are represented by yellow boxes and services (in this case, the Object Transfer Service) are represented by orange boxes.

An Object Server shall instantiate the Object Transfer Service [1]. As shown above, a higher layer specification is required to provide additional requirements on top of OTS/OTP that are specific to the needs of the use case.

2.3 Concurrency Limitations and Restrictions

There are no concurrency limitations or restrictions for the Object Client or Object Server imposed by this profile. A Concurrency feature is specified in the Object Transfer Service [1].

2.4 Topology Limitations and Restrictions

There are no topology limitations or restrictions imposed by this specification when used with either the LE transport or the BR/EDR transport.

The GAP roles for the Object Server and Object Client are to be defined by a higher layer specification.

2.5 Transport Dependencies

There are no transport restrictions imposed by this profile specification. However, a higher layer specification may impose additional requirements.

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

2.6 ASCII Control Characters

Where the term “ASCII control characters” is used in this specification it means the DEL (delete or backspace) character (0x7F), and the characters in the C0 set (0x00 to 0x1F) and C1 set (0x80 to 0x9F) as defined in [5].

3 Object Server Role Requirements

The Object Server shall instantiate one or more Object Transfer Services [1].

A higher layer specification may impose additional requirements on the service declaration of the Object Transfer Service (e.g., specifying whether it is a «Secondary Service» or a «Primary Service») as well as other additional requirements.

There shall not be more than one instance of the Object Transfer Service that is declared as a «Primary Service».

Service	Object Server
Object Transfer Service	M

Table 3.1: *Service Requirements for Object Server*

3.1 Additional Requirements for Low Energy Transport

This section describes additional Object Server requirements beyond those defined in the Object Transfer Service when using this profile over Low Energy transport.

3.1.1 Service UUIDs AD Type

While in a GAP Discoverable Mode for initial connection to an Object Client, the Object Server may include the Object Transfer Service UUID defined in [4] in the Service UUIDs AD type field of the advertising data. This enhances the user experience since an Object Server may be identified by the Object Client before initiating a connection.

3.1.2 Object Transfer Channel

The L2CAP LE Credit Based Flow Control Mode shall be supported.

3.2 Additional Requirements for BR/EDR Transport

This section describes additional Object Server requirements beyond those defined in the Object Transfer Service when using this profile over the BR/EDR transport.

3.2.1 Object Transfer Channel

The L2CAP Enhanced Retransmission Mode shall be supported.

4 Object Client Role Requirements

Table 4.1 describes the discovery requirements for an Object Client.

Discovery Requirement	Section	Support in Object Client
Object Transfer Service Discovery	4.2	M
Object Transfer Service Characteristic Discovery	4.3.1	M

Table 4.1: *Discovery Requirements for Object Client*

Table 4.2 describes the characteristic support requirements for an Object Client.

Characteristic Support Requirements	Support in Object Client
OTS Feature	M
Object Name	M
Object Type	M
Object Size	C.1
Object First-Created	O
Object Last-Modified	C.2
Object ID	C.3
Object Properties	O
Object Action Control Point (OACP)	M
Object List Control Point (OLCP)	M
Object List Filter	M
Object Changed	O

Table 4.2: *Characteristic Support Requirements for Object Client*

C.1: Mandatory if the Read Object Contents sub-procedure (Section 4.5.4.2) or Write Object Contents sub-procedure (Section 4.5.5.2) are supported; otherwise optional.
C.2: Mandatory, if the Write Object Contents sub-procedure (Section 4.5.5.2) is supported; otherwise optional.
C.3: Mandatory, if the Select by Object ID sub-procedure (Section 4.5.3.2) is supported; otherwise optional.

4.1 GATT Sub-Procedure Requirements

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

The table below summarizes *additional* GATT sub-procedure requirements beyond those required by all GATT clients.

GATT Sub-Procedure	Object Client Requirements
Discover All Primary Services	C.1
Discover Primary Services by Service UUID	C.1
Find Included Services	O
Discover All Characteristics of a Service	C.2
Discover Characteristics by UUID	C.2
Discover All Characteristic Descriptors	M
Read Characteristic Value	M
Read Long Characteristic Values	M
Write Characteristic Value	M
Write Long Characteristic Values	O
Read Characteristic Descriptors	M
Write Characteristic Descriptors	M

Table 4.3: *Additional GATT Sub-Procedure Requirements*

C.1: Mandatory to support at least one of these Service Discovery sub-procedures when using the LE transport. Both are Excluded if the LE transport is not supported since SDP must be used over BR/EDR.
C.2: Mandatory to support at least one of these Characteristic Discovery sub-procedures.

4.2 Service Discovery

The Object Client shall discover the Object Transfer Service.

When using the Low Energy transport and performing primary service discovery, the Object Client shall use either the GATT *Discover All Primary Services* sub-procedure or the GATT *Discover Primary Services by Service UUID* sub-procedure.

When using the Low Energy transport and performing secondary service discovery, the Object Client shall use the GATT *Find Included Services* sub-procedure to discover the Object Transfer Service.

When using the BR/EDR transport, the Object Client shall initiate service discovery by retrieving the SDP record of the Object Transfer Service as defined in [1].

4.3 Characteristic Discovery

As required by GATT, the Object Client shall be tolerant of additional optional characteristics in the service records of services used with this profile.

Where a characteristic is discovered that can be indicated or notified, the Object Client shall also discover the associated *Client Characteristic Configuration* descriptor.

4.3.1 Object Transfer Service Characteristic Discovery

The Object Client shall use either the GATT *Discover All Characteristics of a Service* sub-procedure or the GATT *Discover Characteristics by UUID* sub-procedure to discover the characteristics of the service.

The Object Client shall use the GATT *Discover All Characteristic Descriptors* sub-procedure to discover the characteristic descriptors.

4.4 Object Transfer Service Characteristics

This section contains characteristics and behavior that is otherwise not defined in the object transfer procedures in Section 4.5.

4.4.1 Object Action Control Point (OACP)

Before writing to the OACP, the Object Client shall configure the OACP characteristic for indications (i.e., via the *Client Characteristic Configuration* descriptor).

The Object Client may perform a write to the OACP to request a desired Op Code. The Object Server normally responds to each successful write with an indication that includes the Response Code, the Requested Op Code, and a Result Code and may also include a Response Parameter as defined in [1]. A control point timeout process is specified in Section 4.4.4.

The Object Client can determine the supported OACP Op Codes of the Object Server and related capabilities by reading the OACP Features field of the OTS Feature characteristic.

4.4.1.1 OACP Op Code Requirements

Table 4.4 shows the requirements for the OACP Op Codes in the context of this profile:

OACP Op Code	Requirement
OACP Create	Mandatory if the Create Object procedure (Section 4.5.7) is supported; otherwise optional.
OACP Delete	Mandatory if the Delete Object procedure (Section 4.5.8) is supported; otherwise optional.
OACP Calculate Checksum	Mandatory if the Resume Reading Object Contents sub-procedure (Section 4.5.6.1) or the Resume Writing Object Contents - Data Integrity Method sub-procedure (Section 4.5.6.2.2) is supported; otherwise optional.
OACP Execute	Mandatory if the Execute Object procedure (Section 4.5.9) is supported; otherwise optional.
OACP Read	Mandatory if the Read Object Contents sub-procedure (Section 4.5.4.2) or Resume Reading Object Contents sub-procedure (Section 4.5.6.1) are supported; otherwise optional.
OACP Write	Mandatory if the Write Object Contents sub-procedure (Section 4.5.5.2) or any of the Resume Writing Object Contents sub-procedures (Section 4.5.6.2) are supported; otherwise optional.

OACP Op Code	Requirement
OACP Abort	Mandatory if the Read Object Contents sub-procedure (Section 4.5.4.2) is supported; otherwise optional.

Table 4.4: *OACP Op Code Requirements*

The Object Client may write to the OACP characteristic using one of the supported Op Codes in Table 4.4 to request an Object Server to perform an operation. This may include a parameter that is valid within the context of that Op Code as defined in [1].

For all Op Codes described in the following sections, the Object Client shall wait for the OACP Response indication or for the operation to time out according to the process described in Section 4.4.4.

See Section 4.6 for general error handling.

4.4.1.1.1 OACP Create Op Code

To create an object in the Object Server, the Object Client shall use the *OACP Create* Op Code followed by Size and Type parameter values that represent the Allocated Size and Object Type for the object, as defined in [1].

4.4.1.1.2 OACP Delete Op Code

To delete the Current Object in the Object Server, the Object Client shall use the *OACP Delete* Op Code.

4.4.1.1.3 OACP Calculate Checksum Op Code

To calculate the checksum of the Current Object in the Object Server, the Object Client shall use the *OACP Calculate Checksum* Op Code followed by parameter values that represent the Offset and the Length of the object contents to be included in the checksum calculation as defined in [1].

The Response Value will include a parameter that will include the object Checksum.

4.4.1.1.4 OACP Execute Op Code

To execute an operation defined by a higher-level specification, using the Current Object, the Object Client shall use the *OACP Execute* Op Code followed by a parameter value that may also be defined by the higher-level specification.

4.4.1.1.5 OACP Read Op Code

To read the Current Object in the Object Server, the Object Client shall use the *OACP Read* Op Code followed by Parameter values that represent the Offset and Length of the object contents as defined in [1].

4.4.1.1.6 OACP Write Op Code

To write to the Current Object in the Object Server, the Object Client shall use the *OACP Write* Op Code followed by parameter values that represent the Offset and Length of the object contents and the mode as defined in [1].

4.4.1.1.7 OACP Abort Op Code

To request the Object Server to cease sending object data to it, aborting an OACP Read operation in progress, the Object Client shall use the *OACP Abort* Op Code.

4.4.2 Object List Control Point (OLCP)

Before writing to the OLCP, the Object Client shall configure the OLCP characteristic for indications (i.e., via the *Client Characteristic Configuration* descriptor).

The Object Client may perform a write to the OLCP to request a desired Op Code. The Object Server normally responds to each successful write with an indication that includes the Response Code, the Requested Op Code, and a Result Code and may also include a Response Parameter as defined in [1]. A control point timeout process is specified in Section 4.4.4.

The Object Client can determine the supported OLCP Op Codes of the Object Server by reading the OLCP Features field of the OTS Feature characteristic.

4.4.2.1 OLCP Op Code Requirements

Table 4.5 shows the requirements for the OLCP Op Codes in the context of this profile:

Op Code	Requirement
OLCP First	M
OLCP Last	O
OLCP Previous	O
OLCP Next	M
OLCP Go To	C.1
OLCP Order	O
OLCP Request Number of Objects	O
OLCP Clear Marking	O

Table 4.5: *OLCP Op Code Requirements*

C.1: Mandatory if the Select by Object ID procedure (Section 4.5.3.2) is supported; otherwise optional.

The Object Client may write to the OLCP characteristic using one of the supported Op Codes in Table 4.5 to request an Object Server to perform an operation. This may include a parameter that is valid within the context of that Op Code as defined in [1].

For all Op Codes described in the following sections, the Object Client shall wait for the OLCP Response indication or for the operation to time out according to the process described in Section 4.4.4.

See Section 4.6 for general error handling.

4.4.2.1.1 OLCP First Op Code

To select the first object in the Object Server object list based on the present filter and List Sort Order conditions, the Object Client shall use the *OLCP First* Op Code as defined in [1].

4.4.2.1.2 OLCP Last Op Code

To select the last object in the Object Server object list based on the present filter and List Sort Order conditions, the Object Client shall use the *OLCP Last* Op Code as defined in [1].

4.4.2.1.3 OLCP Previous Op Code

To select the previous object in the Object Server object list based on the present filter and List Sort Order conditions, the Object Client shall use the *OLCP Previous* Op Code as defined in [1].

4.4.2.1.4 OLCP Next Op Code

To select the next object in the Object Server object list based on the present filter and List Sort Order conditions, the Object Client shall use the *OLCP Next* Op Code as defined in [1].

4.4.2.1.5 OLCP Go To Op Code

To select a specific object in the Object Server, the Object Client shall use the *OLCP Go To* Op Code followed by a parameter value that represents the Object ID as defined in [1].

4.4.2.1.6 OLCP Order Op Code

To change the sort order of an object list in the Object Server, the Object Client shall use the *OLCP Order* Op Code followed by a parameter value that represents a List Sort Order as defined in [1].

4.4.2.1.7 OLCP Request Number of Objects Op Code

To request the number of objects in the Object Server, the Object Client shall use the *OLCP Request Number of Objects* Op Code.

The Response Value will include a Parameter that will include the Total Number of Objects.

4.4.2.1.8 OLCP Clear Marking Op Code

To clear marked objects in an object list in the Object Server, the Object Client shall use the *OLCP Clear Marking* Op Code. The meaning of a marked object is defined by a higher layer specification.

4.4.3 Object Changed

The requirements in this section only apply if the Object Changed characteristic is supported by the Object Client.

The Object Client may configure indications of the Object Changed characteristic (i.e., via the *Client Characteristic Configuration* descriptor) so that it can be alerted if there are any changes to objects that occur while it is connected that are made:

- at the Object Server (locally)
  or
- by an Object Client other than itself.

This feature is not intended for changes to objects that occur while the Object Client is disconnected. Therefore, upon each connection, Object Clients should confirm that an object of interest is unchanged by using the *OACP Calculate Checksum* Op Code or by reading the Object Last-Modified characteristic and comparing these values to previously cached values.

The Object Client shall be able to receive multiple indications of the Object Changed characteristic from an Object Server for the case where multiple objects have changed.

The Object Client may determine the details of the object change from the contents of the Object Changed characteristic.

The Object Client may use the Object ID field of the Object Changed characteristic to determine which object has been changed.

The Object Client may use the Source of Change bit (bit 0) of the Flags field of the Object Changed characteristic to determine whether the source of the change was at the Object Server or an Object Client other than itself.

Other details may also be determined by the Object Client by decoding bits for:

- Change occurred to the object contents
- Change occurred to the object metadata
- Object Creation
- Object Deletion

If the Object Client receives an Object Changed characteristic indication with Reserved for Future Use (RFU) bits of the Flags field that are non-zero, it shall ignore those bits and any additional data that may be present in the packet and continue to process the Object Changed characteristic as if all the RFU bits had been zero. This is to enable compatibility with future service updates for the case where available octets in the characteristic are specified for optional use. What the Object Client does with the additional, unrecognized octets is left to the implementation.

4.4.4 OACP or OLCP Timeout Process

An OLCP or OACP operation is started when the ATT Write Response is received from the Object Server as a result of the Object Client writing an Op Code to a control point to perform some desired action. The control point operation ends when the Object Server sends an indication to the Object Client with the Op Code set to *Response Code*.

An OLCP or OACP operation is not considered started and not queued in the Object Server when a write to a control point results in an ATT Error Response.

An OACP or OLCP operation is considered to have timed out if a control point indication is not received within a period equal to T_{OACP} or T_{OLCP}, respectively, from the start of the operation. The value for the timers, T_{OACP} and T_{OLCP}, shall each be set to a default of 30 seconds unless otherwise specified by a higher level specification. A higher level specification may define a shorter timeout requirement.

If the link is lost while a control point operation is in progress then the operation shall be considered to have timed out.

Thus, an Object Client shall start a timer with a period set equal to T_{OACP} (for the OACP) or to T_{OLCP} (for the OLCP) after the ATT Write Response is received from the Object Server. The timer shall be stopped when a control point indication is received and the Op Code is set to *Response Code*. If the timer expires, then the operation shall be considered to have failed.

If a control point operation times out, then no new control point operation shall be started by the Object Client until a new link is established with the Object Server. To ensure a positive user experience, if a control point operation times out, the Object Client should disconnect the physical link and then reconnect.

4.5 Object Transfer Procedures

The generic procedures defined in this section provide standardized methods of using the features of the Object Transfer Service [1] in specified sequences in order to satisfy common use cases. This profile does not forbid the use of the features of OTS in other ways and a higher layer profile may define such additional procedures. However, wherever a procedure listed in Table 4.6 is supported, the implementation of the procedure shall conform to the requirements of this profile specification.

v10

Table 4.6 shows procedure requirements for this profile:

Procedures	Reference	Requirement
Feature Discovery	4.5.1	M
Object Discovery – Discover All Objects	4.5.2.1	C.1
Object Discovery – Discover by Filter	0	C.1
Object Discovery – Search for Specific Object	4.5.2.3	C.1
Object Discovery – Discover by Directory Listing Object	4.5.2.4	O
Select Object – Select by Object Discovery	4.5.3.1	C.2
Select Object – Select by Object ID	4.5.3.2	C.2
Read Object – Read Object Metadata	4.5.4.1	M
Read Object – Read Object Contents	4.5.4.2	C.3
Write Object – Write Object Metadata	4.5.5.1	C.4
Write Object – Write Object Contents	4.5.5.2	C.3
Resume Reading Object Contents	4.5.6.1	O
Resume Writing Object Contents – Current Size Method	4.5.6.2.1	O
Resume Writing Object Contents – Data Integrity Method	4.5.6.2.2	O
Create Object	4.5.7	O
Delete Object	4.5.8	O
Execute Object	4.5.9	O
Open Object Transfer Channel	4.5.10	M

Table 4.6: *Procedure Requirements*

C.1: Mandatory to support at least one of these Object Discovery sub-procedures.
C.2: Mandatory to support at least one of these Select Object sub-procedures.
C.3: Mandatory to support at least one of the Read Object Contents (Section 4.5.4.2) or Write Object Contents (Section 4.5.5.2) sub-procedures.
C.4: Mandatory if the Create Object procedure (Section 4.5.7) is supported; otherwise optional.

4.5.1 Feature Discovery

This procedure shall be performed at least once before any other Object Transfer Procedure is attempted.

The Object Client shall read the OTS Feature characteristic to determine the supported features of the Object Server in order to understand its capabilities.

Since all bits of this characteristic are static for the lifetime of the device (i.e., static permanently or until Service Changed is indicated), the Object Client may cache this value.

In many cases, this information will allow the Object Client to adapt to the supported features of the Object Server (e.g., this can be used to suppress unsupported features from the UI of the Object Client).

The Object Client should detect whether the Object Server is the type that can store no more than one object by checking whether the OLCP characteristic (see Section 4.4.2) is exposed; if the OLCP is exposed, this indicates that the Object Server is capable of storing multiple objects.

The Object Client can detect the Object Server's support for the GATT Read Long sub-procedure (and hence its support for Object Metadata characteristic values that may exceed the capacity of the ATT_MTU size) as described within the Read Object Metadata sub-procedure in Section 4.5.4.1 rather than via this procedure.

4.5.2 Object Discovery

This procedure is used by an Object Client to discover objects on an Object Server. Once the objects are discovered, additional information about the objects can be accessed using other procedures. There are four sub-procedures that can be used for object discovery: Discover All Objects, Discover by Filter, Search for Specific Object, and Discover by Directory Listing Object.

4.5.2.1 Discover All Objects

This sub-procedure is used by an Object Client to find all the objects available within an instance of the Object Transfer Service that supports the OLCP (see Section 4.4.2).

If the Object List Filter characteristics are exposed by the Object Server, the Object Client shall ensure that all objects pass the filter by writing the value 0x00 ("No Filter") to all three instances of the Object List Filter characteristic. Note that the value of the Object List Filter characteristics is not guaranteed to be persistent between connections.

The OLCP First and OLCP Next Op Codes shall be used to select the first object and to step through the objects to discover the objects available via the service. For each object selected in this way, the Object Client shall read at least one of the Object Metadata characteristics such as the Object Name before moving on to the next object. The Object Client may compile a local list of the objects and associated object metadata for reference.

The sub-procedure is complete when, in response to requesting an OLCP operation, an Error Response is received and the Result Code is set to "No Object" or "Out Of Bounds".

If the Object Client receives an OLCP Error Response with the "No Object" Result Code, there are zero objects available within the Object Server.

If, in response to requesting an OLCP Next operation, the Object Client receives an OLCP Error Response with the “Out Of Bounds” Result Code, all of the objects have already been discovered.

Optionally, if the OLCP Order Op Code is supported, the Object Client may initially set the List Sort Order so that the object discovery process described above occurs in a certain order. The Object Client may perform the Feature Discovery procedure to discover if the Object Server supports the OLCP Order Op Code.

The Object Client shall be tolerant of discovering unknown object types.

4.5.2.2 Discover by Filter

This sub-procedure is used by an Object Client to find objects that pass specified filter conditions that are available within an instance of the Object Transfer Service. This sub-procedure can be used only if the Object Server supports the OLCP (see Section 4.4.2) and the Object List Filter characteristics.

The Object Client shall write the desired combination of filter conditions to the three Object List Filter characteristics. Note that the values of the Object List Filter characteristics are not guaranteed to be persistent between connections. When writing UTF-8 Filter descriptions (i.e., Name Starts With, Name Ends With, Name Contains, Name is Exactly), the name written by the Object Client shall not include any ASCII control characters as defined in Section 2.6.

A long filter condition value, such as one that contains a long UTF-8 string, may exceed the capacity of the negotiated ATT_MTU size. If the Object Client supports the GATT Write Long sub-procedure and the characteristic value that the Object Client requires to write exceeds (ATT_MTU – 3) octets, the Object Client should try using the GATT Write Long sub-procedure to write the entire value. If the GATT Write Long sub-procedure is requested by the Object Client and the *Request Not Supported* ATT Error response is returned by the Object Server, the Object Client may choose a shorter string value that can be written using the simple GATT Write sub-procedure. If the Object Client has already established that the Object Server does not support the GATT Write Long sub-procedure by receiving a *Request Not Supported* response as described above, the Object Client should not try to use the GATT Write Long sub-procedure on subsequent occasions with the same Object Server.

The OLCP First and OLCP Next Op Codes shall be used to select the first object and to step through the objects to discover the objects available via the service that conform to the specified filter conditions. For each object selected in this way, the Object Client shall read at least one of the Object Metadata characteristics such as the Object Name before moving on to the next object. The Object Client may compile a local list of the objects and associated object metadata for reference.

The sub-procedure is complete when, in response to requesting an OLCP operation, an OLCP Error Response is received and the Result Code is set to “No Object” or “Out Of Bounds”.

If the Object Client receives an OLCP Error Response with the “No Object” Result Code, there is no object to discover on the Object Server that meets the specified filter conditions.

If, in response to requesting an OLCP Next operation, the Object Client receives an OLCP Error Response with the “Out Of Bounds” Result Code, all of the objects that meet the specified filter conditions have already been discovered.

Optionally, if the OLCP Order Op Code is supported, the Object Client may initially set the List Sort Order so that object discovery process described above occurs in a certain order. The Object Client may perform the Feature Discovery procedure to discover if the Object Server supports the OLCP Order procedure.

4.5.2.3 Search for Specific Object

This sub-procedure is used by an Object Client to find a specific object.

The Object Client shall use either the Discover All Objects sub-procedure (Section 4.5.2.1) or the Discover by Filter sub-procedure (Section 0) with the exception that the Object Client shall end the search as soon as the target object is found, thus leaving the target object as the Current Object.

The Object Client shall identify the object by reading its metadata, such as the Object Name. The sub-procedure is complete when either the target object has been found or an OLCP Error Response has been received.

If the Object Client receives an OLCP Error Response, the target of the search was not found.

4.5.2.4 Discover by Directory Listing Object

This sub-procedure is used by an Object Client to find all the objects available within an instance of the Object Transfer Service as an alternative to the Discover All Objects sub-procedure, if the Object Server supports the Directory Listing Object described in [1].

The Object Client shall use the Select by Object ID sub-procedure (Section 4.5.3.2) with the parameter value of the OLCP *Go To* Op Code set to the Object ID of the Directory Listing Object (i.e., 0x000000000000).

Once the Directory Listing Object has become the Current Object, the Object Client shall read the value of the Current Size field of the Object Size characteristic. Following this, the Object Client shall use the OACP Read Op Code with the Offset parameter set to 0 and the Length parameter set to the value it obtained from the Current Size field.

After reading the contents of the Directory Listing Object, the Object Client should cache the information such that is has a complete local listing of the objects available on the Server and their associated object metadata for reference.

4.5.3 Select Object

This procedure is used by an Object Client to select an object on an Object Server to be the Current Object within an instance of the Object Transfer Service. While an object is selected as the Current Object, information about the object can be accessed and an object transfer can be

initiated using other procedures. There are two sub-procedures that can be used to select an object: Select by Object Discovery and Select by Object ID.

Note that since object selection is not guaranteed to be persistent between connections, the Object Client shall select the object it requires upon each reconnection.

4.5.3.1 Select by Object Discovery

This sub-procedure is used to select an object when the Object ID is not (yet) known or the *OLCP Go To* Op Code is not supported.

The Object Client performs the Search for Specific Object sub-procedure (Section 4.5.2.3) to find the target object.

If the Search for Specific Object sub-procedure completes successfully, the target object is now the Current Object. However, if the Search for Specific Object sub-procedure completes with an OLCP Error Response, the object is not available on the Object Server.

4.5.3.2 Select by Object ID

This sub-procedure is used to select an object when the Object ID is known. This sub-procedure can be used only if the Object Server supports the *OLCP Go To* Op Code.

The Object Client shall use the *OLCP Go To* Op Code with the parameter value set to the Object ID of the desired object.

The sub-procedure is complete when, in response to requesting the OLCP Go To operation, an OLCP response is received and the Result Code is set to “Success” or an error value.

If the Object Client receives an OLCP response with the “Success” Result Code, the object with the requested Object ID is now the Current Object.

If the Object Client receives an OLCP Error Response with the “Object ID Not Found” Result Code, an object with the requested Object ID does not exist within the service.

If the Object Client receives an OLCP Error Response with the “No Object” Result Code, there are zero objects available within the service.

Note that the values of the Object List Filter characteristics, if supported, are ignored by the OLCP Go To operation so that they have no effect on this sub-procedure. In addition, if the OLCP response to the OLCP Go To operation is “Success”, the Object Server will also automatically update the value of all the Object List Filter characteristics, if supported, to set the filter type to “No Filter”, for the benefit of subsequent procedures.

4.5.4 Read Object

This procedure is used to read object metadata and object contents from an Object Server. There are two sub-procedures that can be used to read object data: Read Object Metadata and Read Object Contents.

These sub-procedures can be used to copy the Current Object from the Object Server to the Object Client. The Object Client should read both the object contents and the relevant object metadata in order to build a copy of the Current Object with valid metadata.

Before using either of these sub-procedures, the Object Client shall ensure that the relevant object has been selected as the Current Object. If the Object Server stores only a single object (as determined via the Feature Discovery procedure), the single object will be preselected by the Object Server; otherwise, the Select Object procedure may be used.

4.5.4.1 Read Object Metadata

This sub-procedure is used to read the object metadata of the Current Object from the Object Server.

The Object Client shall read the object metadata that it requires from the Object Metadata characteristics exposed by the Object Server.

The Object Client should read the Object Properties characteristic in order to check that the object's properties permit the required operations.

The Object Metadata characteristics that may be read, if exposed by the Object Server, are defined in the Object Transfer Service [1].

However, the Object Client may ignore any Object Metadata characteristic exposed by the Object Server whose information is not required.

Certain Object Metadata characteristics such as the Object Name characteristic have a variable length and the value may exceed the capacity of the negotiated ATT_MTU size. Therefore, if the value read using a simple GATT Read Request has the maximum length of (ATT_MTU – 1) octets, the Object Client should then try using the GATT Read Long sub-procedure to read the rest of the value, in case there are further octets to read.

If the GATT Read Long sub-procedure is requested by the Object Client and the *Request Not Supported* ATT Error response is returned, it indicates that the Object Server does not support object names that are longer than the capacity of the default ATT_MTU size. If an ATT Error Response with the Error Code set to *Attribute Not Long* is returned, this means that the GATT Read Long sub-procedure is supported but there was no more data to read. In either of these cases, the value already read using the simple GATT Read sub-procedure must be the complete value.

If the Object Client has established that the Object Server does not support the GATT Read Long sub-procedure by receiving a *Request Not Supported* response as described above, the Object Client should not try to use the GATT Read Long sub-procedure on subsequent occasions with the same Object Server.

If the Object Client reads the Object Type characteristic it shall detect whether an Object Type UUID using the 16-bit format (2 octets) or 128-bit format (16 octets) has been used by checking the length of the Object Type characteristic value.

The Object Types that may be used may be restricted to specific types in a higher layer specification, as determined by the requirements of the use case.

4.5.4.2 Read Object Contents

This sub-procedure is used to read part or the whole of the contents of the Current Object from the Object Server. This can be used to copy the object contents from the Object Server to the Object Client. This sub-procedure can be used only if the Object Server supports the OACP Read Op Code.

Before performing this sub-procedure, the Object Client should read the Object Properties characteristic and check that the object's properties permit the required operation.

The Object Client shall determine the length of the object contents by reading the value of the Current Size field of the Object Size characteristic.

The Object Client shall open the channel to be used for the object transfer by using the Open Object Transfer Channel procedure, unless the required Object Transfer Channel is already open.

The Object Client shall use the OACP Read Op Code with the Offset value set to allow the object transfer to begin from the desired octet number. For example, to start at the first octet (octet number 0), the Object Client would use an Offset value of 0. In addition, the value of the Length parameter shall be set to the number of octets that are to be transferred. If the entire object contents are to be transferred, the value of the Length parameter shall equal the value obtained from the Current Size field as described above. The Object Client shall use values for Length and Offset so that their sum is less than or equal to the value obtained from the Current Size field as described above. Note that the Object Server will verify that the sum of the values of the Offset parameter and the Length parameter do not exceed the value of the Current Size field of the Object Size characteristic.

Once the expected number of octets have been received, the Object Client may check the integrity of the data received by obtaining a checksum for the same octets from the Object Server by requesting the *OACP Calculate Checksum* Op Code, if supported, keeping the values of the Offset and Length parameters the same as were used with the *OACP Read* Op Code. If the Object Server responds with the OACP Success response, the checksum value will be returned in the Response Parameter. The Object Client may then verify data integrity by comparing the checksum provided by the Object Server with a checksum that it has calculated itself. See also Section 7 for information about the segmentation of the transferred data.

However, if the Object Server closes the Object Transfer Channel or if no object data is received via the Object Transfer Channel or the object data ceases to be received for a continuous period exceeding 30 seconds (or less if a shorter timeout is specified by a higher layer specification), and fewer than the expected number of octets have been received, the Object Client shall

consider the object transfer to have failed and shall write the *OACP Abort* Op Code to the OACP to ensure that the Object Server also considers the transfer to be aborted.

Refer to Section 4.6 for requirements when the Object Transfer Channel is prematurely closed by the Object Server, and to Section 4.5.6.1 for a procedure for resuming the transfer.

Once the Object Client has no further need for the open Object Transfer Channel, it should close the channel.

4.5.5 Write Object

This procedure is used to write object metadata and object contents to an object on an Object Server. There are two sub-procedures that can be used to write object data: Write Object Metadata, and Write Object Contents.

Before using any of these sub-procedures, the Object Client shall ensure that the relevant object has been selected as the Current Object. If the Object Server stores only a single object (determined via the Feature Discovery procedure), the single object will be preselected by the Object Server. Otherwise, the Select Object procedure may be used.

4.5.5.1 Write Object Metadata

This sub-procedure is used to write values to the object metadata of the Current Object on the Object Server. This sub-procedure can be used only in respect of Object Metadata characteristics that the Object Server exposes with the Write property.

The Object Client shall write to one or more of the writable Object Metadata characteristics. The Object Metadata which may be writable includes Object Name, Object First-Created, Object Last-Modified, and Object Properties. Note that once the Object First-Created characteristics is written with a valid time, is not intended to be updated further.

When writing to the Object Name characteristic, the name value written by the Object Client shall not be a zero length string and shall not include any ASCII control characters as defined in Section 2.6.

Certain Object Metadata characteristics such as the Object Name characteristic have a variable length and the value may exceed the capacity of the negotiated ATT_MTU size. If the value that the Object Client requires to write exceeds (ATT_MTU – 3) octets, the Object Client should try using the GATT Write Long sub-procedure to write the entire value.

If the GATT Write Long sub-procedure is requested by the Object Client and the *Request Not Supported* ATT Error response is returned, the Object Client should choose a shorter name that can be written using the simple GATT Write sub-procedure. Note that a variable length characteristic value will automatically be truncated to the new length when the simple GATT Write sub-procedure is used.

If the Object Client has established that the Object Server does not support the GATT Write Long sub-procedure by receiving a *Request Not Supported* response as described above, the Object

Client should not try to use the GATT Write Long sub-procedure on subsequent occasions with the same Object Server.

4.5.5.2 Write Object Contents

This sub-procedure is used to write or overwrite a part or the whole of the contents of the Current Object. Some options within this procedure increase the allocated size of the object (i.e., appending additional data to an object), enable part of the object to be overwritten within the object (i.e., patching an object), and enable an object of a smaller size to be written requiring a truncation of the object. These options are described below. This sub-procedure can be used only if the Object Server supports the *OACP Write* Op Code.

Before performing this sub-procedure, the Object Client should read the Object Properties characteristic and check that the object's properties permit the required operation.

The Object Client shall read the value of the Current Size field of the Object Size characteristic to obtain the current size of the object.

The Object Client shall open the channel to be used for the object transfer by using the Open Object Transfer Channel procedure, unless the required Object Transfer Channel is already open.

For all Write operations, the Offset parameter value shall be less than or equal to the Current Size.

The Object Client shall use the *OACP Write* Op Code with the Offset parameter value set so that writing will begin with the desired octet number. For example, to start at the first octet (octet number 0), the Object Client would use an Offset parameter value of 0. In addition, the value of the Length parameter shall be set to the number of octets that the Object Client intends to write to the object.

If the Appending Additional Data to Objects Supported bit in the OTS Feature characteristic of the Object Server is set to 1, the Object Client may increase the allocated size of the object (i.e., write data past the previous end of an object). This is done by setting the values of the Offset and the Length parameters such that their sum is greater than the Allocated Size field. Since this operation will result in a larger object size, the value of the Allocated Size field and the Current Size field will be updated by the Object Server as described in [1].

When not appending, the sum of the Offset and Length parameter values shall not exceed the value of the Allocated Size field of the Object Size characteristic.

If the Truncation of Objects Supported bit in the OTS Feature characteristic of the Object Server is set to 1, the Object Client may truncate the object (i.e., delete data from a starting point in the object to the end of the object). This is done by setting the values of the Offset and the Length parameters such that their sum is less than the Current Size field of the Object Size characteristic and by setting bit 1 of the Mode Parameter to 1 (truncate). Since this operation will result in a smaller object size, the value of the Allocated Size field and the Current Size field will be updated by the Object Server as described in [1].

If the Patching of Objects Supported bit in the OTS Feature characteristic of the Object Server is set to 1, the Object Client may patch the object (i.e., overwrite a portion within an object). This is done by setting the values of the Offset and the Length parameters such that their sum is less than the Current Size field of the Object Size characteristic and by setting bit 1 of the Mode Parameter to 0 (do not truncate).

When overwriting the entire object, the Offset field shall be set to 0 and the Length field shall be set to the value of the Current Size field.

If the Object Server responds with the OACP Success response, the Object Client shall begin sending the object contents via the open Object Transfer Channel. The Object Client shall transmit the data through the channel in Little Endian format, sending the least significant octet first.

Once the expected number of octets have been sent, the Object Client may check the integrity of the data received by the Object Server by obtaining a checksum for the same octets from the Object Server. This is done by requesting the OACP Calculate Checksum Op Code, if supported, keeping the values of the Offset and Length parameters the same as were used with the OACP Write Op Code. If the Object Server responds with the OACP Success response, the checksum value will be returned in the Response Parameter. The Object Client may then verify the data integrity by comparing the checksum provided by the Object Server with a checksum that it has calculated itself or knows by other means. See also Section 7 for information about the segmentation of the transferred data.

If the Object Last-Modified characteristic is exposed and supports the Write property, the Object Client shall update the value of the Object Last-Modified characteristic when the object is modified. If the Object Client does not have access to a valid date and time to set the object last-modified metadata, the fields of the Object Last-Modified characteristic for which the Object Client does not have valid data shall be written with a 0 to show that valid data is not available.

However, if the Object Server closes the Object Transfer Channel or if no object data can be sent via the Object Transfer Channel for a continuous period exceeding 30 seconds (or less if a shorter timeout is specified by a higher layer specification) and fewer than the required number of octets have been sent, the Object Client shall consider the object transfer to have failed. Refer to Section 4.6 for requirements when the Object Transfer Channel is prematurely closed by the Object Server, and Section 4.5.6.2 for a procedure for resuming the transfer.

Once the Object Client has no further need for the open Object Transfer Channel, it should close the channel.

4.5.6 Resume

This procedure is used to resume an object transfer when the transfer was interrupted prematurely for a reason such as link loss. There are two sub-procedures that can be used to resume an object transfer: Resume Reading Object Contents and Resume Writing Object Contents. See also Section 7 for information about the segmentation of the transferred data.

4.5.6.1 Resume Reading Object Contents

This sub-procedure is used to resume reading object contents from an offset position when a previous object transfer was interrupted. This sub-procedure can be used only if the Object Server supports the *OACP Read* and *OACP Calculate Checksum* Op Codes.

The Object Client shall determine values for the Offset and Length parameters of the *OACP Calculate Checksum* Op Code that, in combination, specify the octets that the Object Client requires to verify.

The Object Client shall write the *OACP Calculate Checksum* Op Code to the OACP, using the values for the Offset and Length parameters that it determined above. If the Object Server responds with the OACP Success response, the checksum value will be returned in the Response Parameter.

The Object Client may then verify the data integrity by comparing the checksum provided by the Object Server with a checksum that it has calculated itself.

Once the Object Client has determined that the octets it has received so far, or a subset of those octets, pass its data integrity checks, the Object Client shall calculate new values for Offset and Length parameters of the *OACP Read* Op Code that, in combination, define the octets that the Object Client still needs to read from the Object Server. These values can then be used within the Read Object Contents procedure to read the octets from the required position in the object contents.

v10

4.5.6.2 Resume Writing Object Contents

There are two methods that may be used to resume writing an interrupted object transfer. The Current Size method reads the Current Size field of the Object Size characteristic to find the actual number of octets successfully written during the interrupted object transfer. The Data Integrity Method examines portions of the written data to test which parts have been successfully written without errors, enabling an iterative approach.

4.5.6.2.1 Current Size Method

This sub-procedure is used to resume writing object contents from an offset position when a previous object transfer was interrupted. This sub-procedure relies on using the dynamically increasing value of the Current Size field of the Object Size characteristic to determine the point that had been reached in writing the object contents when the object transfer was interrupted.

This method shall not be used to resume patching of an object (i.e., when overwriting a portion of data within an object) as patching does not increase the Current Size value; the data integrity method (see Section 4.5.6.2.2) may be used as an alternative.

The Object Client shall read the Current Size field of the Object Size characteristic.

The Object Client may perform additional data integrity checks.

The value of the Offset parameter of the *OACP Write* Op Code shall be set equal to the value obtained from the Current Size field of the Object Size characteristic.

The Object Client shall calculate a new value for the Length parameter of the *OACP Write* Op Code that defines the remaining octets that the Object Client still requires to write. These values can then be used within the Write Object Contents procedure to write the required octets to the Object Server, resuming from the required position within the object contents.

4.5.6.2.2 Data Integrity Method

This sub-procedure may be used to resume writing object contents from an offset position when a previous object transfer was interrupted or to patch corrupted contents. This sub-procedure can be used only if the Object Server supports the *OACP Write* and *OACP Calculate Checksum* Op Codes.

The Object Client can test the data that has been written to the Object Server so far by using the *OACP Calculate Checksum* Op Code. If desired, it can progressively sub-divide the object contents into smaller parts to identify which part(s) of the object contain valid data.

The Object Client shall determine values for the Offset and Length parameters of the *OACP Calculate Checksum* Op Code that, in combination, specify the octets that the Object Client requires to verify.

The Object Client shall write the *OACP Calculate Checksum* Op Code to the OACP, using the values of the Offset and Length parameters that it determined above. If the Object Server responds with the OACP Success response, the checksum value will be returned in the Response Parameter.

The Object Client shall then verify the data integrity by comparing the checksum provided by the Object Server with a checksum that it has calculated itself.

Once the Object Client has determined which octets within the object contents pass or fail its data integrity checks, the Object Client shall calculate new values for Offset and Length parameters of the *OACP Write* Op Code that, in combination, define the octets that the Object Client still needs to write or overwrite. These values can then be used within the Write Object Contents procedure to write the required octets to the Object Server at the required position(s) within the object contents.

4.5.7 Create Object

This procedure is used by an Object Client to create a new object on the Object Server. This procedure can be used only if the Object Server supports the *OACP Create* Op Code. When an object is created, it will initially be an empty, unnamed object that has specific object type and object size metadata but no contents.

The Object Client shall write the *OACP Create* Op Code to the OACP with the Size parameter set to the number of octets that need to be allocated for the object contents and the Type parameter set to the UUID defining the required object type.

If the response to the *Create* Op Code is “Success”, the Object Client shall write a name to the Object Name characteristic. The name written by the Object Client shall not be a zero length string and shall not include any character codes in the ASCII control characters as defined in Section 2.6. Note that if disconnection occurs before the name is written successfully, the Object Server will delete the new object as a malformed object.

If the response to the *Create* Op Code is “Success”, the values of the Object List Filter characteristics, if present, are updated by the Object Server to ensure that the filter type is set to “No Filter” and the newly created object will automatically be designated as the Current Object for the benefit of subsequent procedures.

Once the new object has been created and named, the Object Client should write values to any other writable object metadata and write the object contents to the Object Server by using the Object Write sub-procedures. If the Object Client has access to a valid date or time to set the object first-created and object last-modified metadata, it should write the values to the relevant fields of the Object First-Created and Object Last-Modified characteristics, if supported by the Object Server. However, the value of any field of the Object First-Created and Object Last-Modified characteristics for which the Object Client does not have valid data shall remain set to the default of 0.

If the Object ID characteristic is exposed by the Object Server, the Object Client should read the Object ID characteristic value, using the Read Object Metadata sub-procedure, in order to discover the dynamically allocated Object ID that has been granted to the new object.

v10

4.5.8 Delete Object

This procedure is used by an Object Client to delete an object from the Object Server. This procedure can be used only if the Object Server supports the OACP Delete Op Code.

The Object Client shall ensure that the object that it intends to delete from the Object Server has been selected as the Current Object. If the Object Server stores only a single object (as determined via the Feature Discovery procedure), the single object will be preselected by the Object Server; otherwise, the Select Object procedure may be used.

Before performing this procedure, the Object Client should read the Object Properties characteristic and check that the object’s properties permit deletion.

The Object Client shall write the *OACP Delete* Op Code to the OACP.

If the response to the *Delete* Op Code is “Success”, the object has been deleted from the Object Server.

When an object is deleted, it will no longer be possible to perform any further actions using that object or its metadata. Note that the Object Server will automatically consider the Current Object will be invalid.

4.5.9 Execute Object

This procedure is used by an Object Client to execute an object that is present on the Object Server. This procedure can be used only if the Object Server supports the *OACP Execute* Op Code. Executing an object is equivalent to ‘double-clicking’ or ‘double-tapping’ an object in the graphical user interface of typical operating systems. Doing so executes the function associated with that object. If the object itself is not an executable, it may typically be passed to an application. Therefore, the action that is initiated by executing an object depends on the object type and is defined in the object type definition.

The Object Client shall ensure that the object that it intends to execute has been selected as the Current Object. If the Object Server stores only a single object (as determined via the Feature Discovery procedure), the single object will be preselected by the Object Server; otherwise, the Select Object procedure may be used.

Before performing this procedure, the Object Client should read the Object Type characteristic in order to determine how the Object Server will interpret the instruction to execute the object, since the resulting behavior is determined by the Object Type.

The Object Client should also read the Object Properties characteristic to check that the object’s properties permit execution.

The Object Client shall write the *OACP Execute* Op Code to the OACP.

If the response to the *Execute* Op Code is “Success”, the request to execute the object has been accepted by the Object Server.

4.5.10 Open Object Transfer Channel

This procedure is used to open an object transfer channel in preparation for the transfer of object contents in either direction. There are two sub-procedures and these depend on the transport to be used for the object transfer: LE Transport or BR/EDR Transport.

The Protocol Service Multiplexer (PSM) value «PSM_OTS» used below represents the fixed PSM value for the Object Transfer protocol, as defined in [4].

For the LE transport, the Object Client shall use the fixed PSM value «PSM_OTS» to compose a LE Credit Based Connection Request which it shall issue in order to open the required LE L2CAP connection oriented channel (available in Bluetooth Core Specification v4.1 or later [3]).

For the BR/EDR transport, the Object Client shall use the fixed PSM value «PSM_OTS» to compose a L2CAP Connection Request which it shall issue in order to open the required L2CAP connection oriented channel with Enhanced Retransmission mode.

This channel that is opened is referred to generically as the Object Transfer Channel.

The transport that is used for sending the OACP request that requires an object transfer to be performed determines the transport that shall be used for the associated Object Transfer Channel;

the Object Client shall use the same transport (e.g., LE) for the Object Transfer Channel as is used for the GATT transaction.

Once an Object Transfer Channel has been successfully opened, it may be used for one or more object transfer operations initiated by the Object Client.

If an Object Transfer Channel has not been successfully opened (e.g., if the LE Credit Based Connection Request or L2CAP Connection Request was refused), the Object Client should not attempt to use the OACP Read or OACP Write Op Codes.

When the Object Client has no more objects to send or receive, it should close the Object Transfer Channel.

4.6 General Error Handling

The Object Client shall be tolerant and behave appropriately (i.e., the Object Client shall be able to continue to process commands and/or receive data normally) when receiving the Result Codes, ATT Application Error Codes, and ATT Error Codes defined in the Object Transfer Service [1].

The Object Client shall be tolerant and behave appropriately (i.e., the Object Client shall be able to continue to process commands and/or receive data normally) if the Object Server closes the Object Transfer Channel during a Read or Write procedure as described in Section 4.5.4.2 or Section 4.5.5.2, respectively.

If a Service Changed indication is received from the Object Server, this indicates not only that the Object Client shall re-perform Service and Characteristic discovery (as defined in GATT [2]) within the handle range specified, but also that the cached values for characteristics and descriptors may no longer be valid and the Object Client is required to refresh these cached values. The Object Client should also re-perform the Select by Object ID sub-procedure or Select by Object Discovery sub-procedure, since any previous selection may no longer be valid and the Current Object may have changed.

If the Object Client reads a characteristic with Reserved for Future Use (RFU) bits that are non-zero (i.e., RFU bits in the OTS Feature characteristic or the Object Properties characteristic), it shall ignore those bits and continue to process the characteristic as if all the RFU bits had been zero. Similarly, if the Object Client reads a characteristic with an enumerated value that is RFU (i.e., RFU values in the Object List Filter characteristic), it shall ignore that value and continue to process commands and to operate normally. This is to enable compatibility with future Object Transfer Service updates.

5 Connection Establishment Procedures

Connection establishment requirements may be defined by the higher layer specification.

6 Security Considerations

Security requirements may be defined by the higher layer specification.

7 Segmentation of Object Data

The object data to be transferred may need to be segmented so that one L2CAP Service Data Unit (SDU) spans more than one information frame.

For the LE transport, the L2CAP SDU Length field is always included in the first LE-frame of the SDU. All subsequent LE-frames that are part of the same SDU do not contain the L2CAP SDU Length field.

For the BR/EDR transport, whether or not the L2CAP SDU Length field is present in an information frame is known from the value of the Segmentation and Reassembly (SAR) field provided in the header.

For either transport, the L2CAP SDU Length field consists of two octets whose value represents the aggregate length of the SDU across all the segments. The available capacity for the information payload is therefore reduced by two octets when the L2CAP SDU Length field is included in the information frame.

8 Generic Access Profile for BR/EDR

Generic Access Profile requirements for BR/EDR are defined by the higher layer specification.

8.1 L2CAP Interoperability Requirements

8.1.1 Flow and Error Control Option

The L2CAP connection-oriented channel for the Object Transfer Channel shall be configured to use Enhanced Retransmission Mode (ERTM).

8.1.2 Frame Checksum (FCS) Option

The value of the FCS option should be configured to the default value of “16-bit FCS”.

9 Acronyms and Abbreviations

Abbreviation or Acronym	Meaning
AD	Advertising Data
AMP	Alternate MAC/PHY
BR/EDR	Basic Rate / Enhanced Data Rate
GAP	Generic Access Profile
GATT	Generic Attribute Profile
IEEE	Institute of Electrical and Electronics Engineers
LE	Low Energy
MTU	Maximum Transmission Unit
OTP	Object Transfer Profile
OTS	Object Transfer Service
PSM	Protocol Service Multiplexer
RFU	Reserved for Future Use
SAR	Segmentation and Reassembly
SDU	Service Data Unit
SDP	Service Discovery Protocol
UI	User Interface
UTF	Unicode Transformation Format
UUID	Universally Unique Identifier

Table 9.1: *Abbreviations and Acronyms*

10 References

[1] Object Transfer Service v1.0.

[2] Bluetooth Core Specification v4.0 or later version of the Core Specification.

[3] Bluetooth Core Specification v4.1 or later version of the Core Specification.

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

[5] ISO/IEC 6429: 1992, Information Technology - Control functions for coded character sets, 3rd ed.