ByteIO Interoperability Testing Specification

Status of This Memo

This memo provides information to the Grid community on the terms and definition of interoperability that apply to implementations of the ByteIO specification. Distribution is unlimited.

Trademarks

OGSA is a trademark of the Open Grid Forum.

Abstract

This document defines the process and support information necessary to perform Interoperability Testing of any two implementations.

The information gathered following the information given in this document is meant to be fed into an Experiences Document that is required for any Recommended Standard on OGF to advanced in becoming a Published Standard.

Contents

Abstract 1
1. Introduction 4
1.1 Goals 4
1.2 Non-Goals 4
1.3 Outline for this Document 4
1.4 Terminology 4
1.5 Namespaces 5
2. Interoperability Foundations 5
2.1 Transport Protocol binding 5
2.2 Factory operations 5
2.3 Test case execution 6
2.4 Security considerations 7
3. Interoperability Factory Interface 7
3.1 Random ByteIO 7
3.2 Streamable ByteIO Factory 8
3.2.1 Seekable resources 8
3.3 Resource contents 8
4. Random ByteIO Interoperability Tests 9
4.1 wsrf-rp:GetResourceProperty operation 9
4.1.1 Interoperability requirements 9
4.2 rbyteio:read operation (i) 9
4.2.1 Interoperability requirements 10
4.3 rbyteio:read operation (ii) 10
4.3.1 Interoperability requirements 11
4.4 rbyteio:read operation (iii) 11
4.4.1 Interoperability requirements 12
4.5 rbyteio:write operation (i) 12
4.5.1 Invoking the operation rbyteio:write() 13
4.5.2 Invoking the operation rbyteio:read() 13
4.5.3 Interoperability requirements 13
4.6 rbyteio:write operation (ii) 14
4.6.1 Invoking the operation rbyteio:write() 14
4.6.2 Invoking the operation rbyteio:read() 14
4.6.3 Interoperability requirements 15
4.7 rbyteio:write operation (iii) 15
4.7.1 Invoking the operation rbyteio:write() 15
4.7.2 Invoking the operation rbyteio:read() 16
4.7.3 Interoperability requirements 16
4.8 rbyteio:append operation 17
4.8.1 Invoking the operation rbyteio:write() 17
4.8.2 Invoking the operation rbyteio:read() 17
4.8.3 Interoperability requirements 17
4.9 rbyteio:truncAppend operation 18
4.9.1 Invoking the operation rbyteio:write() 18
4.9.2 Invoking the operation rbyteio:read() 18
4.9.3 Interoperability requirements 19
5. Streamable ByteIO Interoperability Tests 19
5.1 wsrf-rp:GetResourceProperty operation 19
5.1.1 Interoperability requirements 19
5.2 sbyteio:seekRead operation 20
5.2.1 Interoperability requirements 20
5.3 sbyteio:seekWrite operation 21
5.3.1 Invoking the operation sbyteio:seekWrite() 21
5.3.2 Invoking the operation sbyteio:seekRead() 21

5.3.3 Interoperability requirements 22
Author Information 22
Acknowledgements 22
Intellectual Property Statement 22
Full Copyright Notice 23
References 23
Appendix A Normative Interoperability Interface 24
Appendix B Normative RandomByteIO Binding for Interoperability Tests 27
Appendix C Normative StreamableByteIO Binding for Interoperability Tests 32

1. Introduction

The ByteIO specification defines two interfaces for POSIX-like access to remote resources in a Grid environment:

- RandomByteIO interface for positional random access to resources, and
- StreamableByteIO interface for sequential access to resources.

OGSA WSRF Basic Profile [OGSA BP 1.0] compliant renderings are also normatively specified. However, both renderings follow the style of "short WSDL", that is, data types and port types are defined in WSDL, but the binding to a particular transport protocol, such as SOAP 1.1 or SOAP 1.2 are left out. While this has certain benefits for implementers (i.e. implementations are easier to embed into existing Grid infrastructures) it has consequences on the interoperability of any two ByteIO implementations.

1.1 Goals

This document defines the Interoperability Tests for the ByteIO Working Group. Interoperability Tests in a Web Service environment ensure that two implementations (typically a client and a server implementation) understand the messages that are exchanged. Explicitly, Interoperability Tests do not test for correctness of values carried in messages.

Therefore, this document defines tests that ensure that messages, whether request or response messages that are formatted according to the ByteIO specification, can be properly processed by client and server implementations. While the test definitions contain concrete parameter values that must be met in order to pass the test, the primary focus of Interoperability Tests lie on testing the semantics of the invoked operation. In that respect, Interoperability Tests are also unit tests, which is by any means unintentional.

1.2 Non-Goals

This document does not focus on formal parameter value correctness. Hence this document does not cover border cases, which are the domain of unit tests.

1.3 Outline for this Document

The remainder of this document will be organized as follows. Section 2 defines the necessary Interoperability foundations to enable interoperability for ByteIO implementations. Section 3 defines the Resource Factory that every Interoperability implementation must support. Sections 4 and 5 define the Interoperability Tests for the Random ByteIO and Streamable ByteIO specification, respectively. Section 6 defines compliance levels an implementation can claim, and the process of advertising this information.

1.4 Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

When describing concrete XML schemas, this specification uses the notational convention of [**WS-Security**]. Specifically, each member of an element's [children] or [attributes] property is described using an XPath-like notation (e.g., /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard (<xsd:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xsd:anyAttribute/>).

1.5 Namespaces

The following namespaces are used in this document:

Prefix	Namespace
byteio	http://schemas.ggf.org/byteio/2005/10/byte-io
rbyteio	http://schemas.ggf.org/byteio/2005/10/random-access
sbyteio	http://schemas.ggf.org/byteio/2005/10/streamable-access
interop	http://schemas.ggf.org/byteio/2006/07/interop
wsrf-rp	http://docs.oasis-open.org/wsrf/rp-2
wsa	http://www.w3.org/2005/08/addressing

2. Interoperability Foundations

To allow for interoperability, several issues need to be addressed to allow any two ByteIO implementations to communicate with each other. The following sub-sections define the framework that each ByteIO implementation MUST follow to enter interoperability tests.

2.1 Transport Protocol binding

The ByteIO WSDL renderings for the Random ByteIO and Streamable ByteIO specifications follow the style of “short WSDL” to allow flexibility in embedding the implementations into existing Grid frameworks. However, interoperability in implementations requires agreement on a specific Transport Protocol binding.

ByteIO implementations that wish to participate in interoperability tests MUST use the following binding:

- The Port Type MUST be bound to SOAP 1.1
- The binding MUST use the “document/literal” style.

The normative specification of the bindings for Random Access ByteIO and Streamable Access ByteIO are given in Appendix B and Appendix C, respectively.

2.2 Factory operations

In OGSA, factory operations are out of scope. Web Service endpoints (in WSRF: WS Resources) are assumed to be either present from the beginning, or obtained otherwise, e.g. registries, directories and other discovery mechanisms.

However, the lack of a standardised means to obtain WS endpoints makes interoperability testing rather difficult. To enable interoperability testing in ByteIO, this document defines a complete factory interface (including a binding) that each ByteIO implementation that wishes to participate in interoperability tests must support.

ByteIO implementations that wish to participate in interoperability tests MUST support the factory interface defined in this section. The concrete means of support is left to the implementations:
Implementations MAY implement the normative interface similar to how they implement the ByteIO specifications. Alternatively, implementations MAY choose to directly consume the messages that are defined by the normative factory interface.

2.3 Test case execution

This document inherits the principles of testing from unit testing frameworks such as JUnit [] and NUnit [], and experiences from WSRF interoperability workshops.

Although JUnit and NUnit are targeted at unit testing (i.e. validating that an Java or .NET class implements an interface or a method signature in an expected manner), the general approach to test case execution still applies to Web Services interoperability testing.

WSRF interoperability workshops took the approach of having the client invoke the operations and analyse the outcome. If the invocation of the operation involves state change on the tested server (e.g. write operations) then the client explicitly invokes an appropriate operation that allows it to analyse the result on the server and assess whether the interoperability test was successful.

This Interoperability Specification takes a hybrid approach combining the principles of unit testing with the WSRF Interoperability approach: Each test execution includes the set up and tear down phases of unit testing, and the phase of test operation execution, necessary reading operation invocations are added where appropriate.

When invoking a subsequent read operation is necessary (e.g. such as in section 4.5) its description is given in non-normative manner but the essential outcome of that operation is given as normative text.

In detail, each test case specified later in this document must be carried out as follows:

The ByteIO interoperability client MUST invoke the interoperability factory operation “interop:CreateResource” before carrying out the Interoperability Test Description.
The ByteIO interoperability client MUST carry out the Interoperability Test Description.
The ByteIO interoperability client MUST invoke the interoperability factory operation “interop:DeleteResource” after carrying out the Interoperability Test Description.

The following illustrates this non-normatively in greater detail:

1. The ByteIO interoperability client invokes the operation interop:CreateResource using a predefined contents.
    a. The ByteIO server creates the resource with the given contents
    b. The ByteIO server mints an EPR and hands it back to the ByteIO client.

2. The ByteIO client invokes the method that is tested in the designated Interoperability Test description.
    a. The ByteIO server executes the invoked operation, and returns the response.
    b. If applicable, the ByteIO client invokes an appropriate operation, such as rbyteio:read(), to examine any changes that have been made on the server.

3. The ByteIO interoperability client invokes the operation interop:DeleteResource
    a. The ByteIO Server destroys the resource; later invocations on the same EPR MUST result in Resource Unavailable Faults.

Steps 1 and 3 correspond to the JUnit TestCase class methods “setUp()” and “teardown()”, respectively.

2.4 Security considerations

To protect exposed resources and access credentials, ByteIO implementations need to secure connections between clients and servers. How this is done is left to the implementers.

Regarding testing interoperability security considerations can be relaxed to the extent that no security at all is required in this special situation.

Testing interoperability as per this document does not touch or involve real resources at the server, given that every server implementation exactly follows the instruction given in this document. For each test case execution an artificial resource is created at the server (through the interoperability factory port type) and an EPR to that resource is returned. This EPR could be specially minted so that the ByteIO server implementation would recognise and relax security constraints for this resource. For example, a ByteIO implementation could make use of the [reference parameter] section of such EPR.

Having said that, the ByteIO Interoperability Specification defines the following security policy for Interoperability Testing Events:

During Interoperability Testing, server and client

- MUST NOT communicate over SSL or TLS secured channels,
- SHOULD NOT add security information or elements to the SOAP messages passed back and forth
- MUST add the parameter “mustUnderstand=”false”” to any security information that is added to any SOAP message during interoperability testing
- MUST NOT expect any security information in the SOAP messages passed back and forth during interoperability testing

---

3. Interoperability Factory Interface

The ByteIO WG defines an interface that is used to initialize the environment for each test. Appendix A normatively defines this interface. ByteIO implementations, whether client or server, that claim interoperability must follow the requirements given in chapter 0.

This Factory Interface is used to test both a Random ByteIO implementation, and a Streamable ByteIO implementation. However, the contents and semantics of the “interop:CreateResource” operation differ slightly for the invocations for a Random ByteIO or Streamable ByteIO implementation. The method “interop:DestroyResource” has the same invariant semantics whatsoever.

Furthermore, a ByteIO server MUST create and maintain different resources for each interop:CreateResource operation invocation.

3.1 Random ByteIO

To test an implementation of the Random ByteIO interface, a client MUST invoke the interop:CreateResource operation (or emit a message that is compatible to that operation) prior to invoking the Random ByteIO operation.

A Random ByteIO server MUST be able to receive such message. When receiving such message, the server MUST create a resource (e.g. a file on the server’s local file system) and an

appropriate EPR. The server MUST respond by sending back the appropriate message to the client.

3.2 Streamable ByteIO Factory

To test an implementation of the Streamable ByteIO interface, a client MUST invoke the interop:CreateResource operation (or emit a message that is compatible to that operation) prior to invoking the Streamable ByteIO operation.

A Streamable ByteIO server MUST be able to receive such message. When receiving such message, the server MUST create a resource. The resource MUST NOT deliver any more bytes once the end of stream has been reached.

The resource must be created so that the current stream position is at the beginning of that resource, i.e. “0”.

3.2.1 Seekable resources

As outlined in section 2.3 test cases that involve any manipulation of the resource on the server must include means to check that this manipulation has been carried out in the desired manner. Typically, such situation occurs when the invocation of a write operation of some sort is tested for interoperability; the tested write operation is then followed by an appropriate read operation to retriefe the up-to-date contents of the remote resource.

To ensure that interoperability can be tested for Streamable ByteIO implementations, the tested server side implementation MUST expose the test resource as a seekable resource as per definition in the ByteIO Specifcation, section 2.4.

The WS Resource that is referenced by the EPR a server side implementation mints on calling the operation “interop:CreateResource” MUST be a seekable stream.

To reduce complexity and possible failures in interoperability testing, the test case will use only one pattern to ensure the desired result of any tested write style operation. For this to work, the interoperability specification defines the following requirements for seekable streams:

The seek operation identified by the URI “http://ggf.org/byte-io/2005/10/streamable-access/seek-origins/beginning” MUST be supported.

The test case(s) then will seek to the beginning of the resource and read the whole contents of the resource.

3.3 Resource contents

Regardless the ByteIO implementation is a Streamable ByteIO or Random ByteIO implementation the contents of the resource is always the same. When the ByteIO client invokes the interop:CreateResource operation, the ByteIO server MUST create the resource and fill it with the content as defined as follows.

The content is an array of 60 bytes. This array, if concatenated and interpreted in US-ASCII encoding, forms a string of one line with 60 characters. These characters form the integer numbers from one to thirty including leading zeros:

“010203040506070809101112131415161718192021222324252627282930”

The following is the normative definition of the contents. Each byte value is given in hexadecimal notation. For example, “0x54 0x68 0x65 0x20 0x61 0x6E 0x73 0x77 0x65 0x72 0x20 0x69 0x73 0x20 0x34 0x32 0x2E” interpreted as given above would form the string “The answer is 42.”.

The sequence of bytes a resource contains after it has been created MUST be as follows, in that given order:
0x30 0x31 0x30 0x32 0x30 0x33 0x30 0x34 0x30 0x35 0x30 0x36 0x30 0x37 0x30 0x38 0x30 0x39 0x31 0x30 0x31 0x31 0x31 0x32 0x31 0x33 0x31 0x34 0x31 0x35 0x31 0x36 0x31 0x37 0x31 0x38 0x31 0x39 0x32 0x30 0x32 0x31 0x32 0x32 0x32 0x33 0x32 0x34 0x32 0x35 0x32 0x36 0x32 0x37 0x32 0x38 0x32 0x39 0x33 0x30

4. Random ByteIO Interoperability Tests

This chapter defines the Interoperability tests for the Random ByteIO interface specification. The tests are defined in a logical order. However, this order of definition does not imply any particular order of execution.

4.1 wsrf-rp:GetResourceProperty operation

The Random ByteIO interoperability client invokes the method “wsrf-rp:GetResourceProperty” to indicate that she wishes to get the value of a given resource property.

In this case, the resource property is “rbyteio:Size”.

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<wsrf-rp:GetResourceProperty> rbyteio:Size </wsrf-rp:GetResourceProperty>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<wsrf-rp:GetResourcePropertyResponse>
   <rbyteio:Size> 60 </rbyteio:Size>
</wsrf-rp:GetResourcePropertyResponse>
```

4.1.1 Interoperability requirements

1. The client MUST use the WSRF RP message “wsrf-rp:GetResourceProperty”
2. The client MUST use the QName “rbyteio:Size” to request the Random ByteIO Size Resource Property.
3. The server MUST use the WSRF RP message “wsrf-rp:GetResourcePropertyResponse”.
4. The server MUST return the value “60” for the Resource Property “rbyteio:Size”.

4.2 rbyteio:read operation (i)

The Random ByteIO interoperability client invokes the method “rbyteio:read” to indicate that she wishes to read a set of bytes from the addressed resource.

In this case, the request describes a consecutive sequence of 12 bytes starting at and including the byte at position 20. Interpreted as a string in US-ASCII, this block of bytes equivalents to “111213141516”.

This request intends to test the general interoperability of the rbyteio:read operation.

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<rbyteio:read>
   <rbyteio:start-offset> 20 </rbyteio:start-offset>
   <rbyteio:bytes-per-block> 6 </rbyteio:bytes-per-block>
   <rbyteio:num-blocks> 1 </rbyteio:num-blocks>
   <rbyteio:stride> 0 </rbyteio:stride>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”/>
</rbyteio:read>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<rbyteio:readResponse>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”>
      MTExMjEzMTQxNTE2
   </rbyteio:transfer-information>
</rbyteio:readResponse>
```

4.2.1 Interoperability requirements

1. The client MUST use the Random ByteIO message “rbyteio:read”.
2. The client MUST use “20” as value for rbyteio:read/rbyteio:start-offset
3. The client MUST use “12” as value for rbyteio:read/rbyteio:bytes-per-block
4. The client MUST use “1” as value for rbyteio:read/rbyteio:num-blocks
5. The client MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:read/rbyteio:transfer-information/@transfer-mechanism
6. The server MUST use the Random ByteIO message “rbyteio:readResponse”.
7. The server MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:readResponse/rbyteio:transfer-information/@transfer-mechanism
8. The server MUST return “MTExMjEzMTQxNTE2” as value for rbyteio:readResponse/rbyteio:transfer-information

4.3 rbyteio:read operation (ii)

The Random ByteIO interoperability client invokes the method “rbyteio:read” to indicate that she wishes to read a set of bytes from the addressed resource.

In this case, the request describes five sequences of two bytes each. These five blocks have a distance of 4 bytes between their respective start positions. The first block starts at and includes the byte at position 2. Interpreted as a string in US-ASCII, the byte array described in this request equivalents to “0204060810”.

This request intends to test correct interpretation of the rbyteio:stride parameter of the rbyteio:read operation, and how the resulting byte array is constructed in case the value for rbyteio:stride exceeds the block size. It also tests repetitive block reading and concatenation with the result byte array.

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<rbyteio:read>
   <rbyteio:start-offset> 2 </rbyteio:start-offset>
   <rbyteio:bytes-per-block> 2 </rbyteio:bytes-per-block>
   <rbyteio:num-blocks> 5 </rbyteio:num-blocks>
   <rbyteio:stride> 4 </rbyteio:stride>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”/>
</rbyteio:read>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<rbyteio:readResponse>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”>
      MDIwNDA2MDgxMA==
   </rbyteio:transfer-information>
</rbyteio:readResponse>
```

4.3.1 Interoperability requirements

1. The client MUST use the Random ByteIO message “rbyteio:read”.
2. The client MUST use “2” as value for rbyteio:read/rbyteio:start-offset
3. The client MUST use “2” as value for rbyteio:read/rbyteio:bytes-per-block
4. The client MUST use “5” as value for rbyteio:read/rbyteio:num-blocks
5. The client MUST use “4” as value for rbyteio:read/rbyteio:stride
6. The client MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:read/rbyteio:transfer-information/@transfer-mechanism
7. The server MUST use the Random ByteIO message “rbyteio:readResponse”.
8. The server MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:readResponse/rbyteio:transfer-information/@transfer-mechanism
9. The server MUST return “MDIwNDA2MDgxMA==” as value for rbyteio:readResponse/rbyteio:transfer-information

4.4 rbyteio:read operation (iii)

The Random ByteIO interoperability client invokes the method “rbyteio:read” to indicate that she wishes to read a set of bytes from the addressed resource.

In this case, the request describes two sequences of 12 bytes each. These two blocks have a distance of 6 bytes between their respective start positions. The first block starts at and includes the byte at position 0. Interpreted as a string in US-ASCII, the byte array described in this request equivalents to “010203040506040506070809”.

This request intends to test correct interpretation of the rbyteio:stride parameter of the rbyteio:read operation, and how the resulting byte array is constructed in case the value for rbyteio:stride is smaller than the block size.

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<rbyteio:read>
   <rbyteio:start-offset> 0 </rbyteio:start-offset>
   <rbyteio:bytes-per-block> 12 </rbyteio:bytes-per-block>
   <rbyteio:num-blocks> 2 </rbyteio:num-blocks>
   <rbyteio:stride> 6 </rbyteio:stride>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”/>
</rbyteio:read>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<rbyteio:readResponse>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”>
      MDEwMjAzMDQwNTA2MDQwNTA2MDcwODA5
   </rbyteio:transfer-information>
</rbyteio:readResponse>
```

4.4.1 Interoperability requirements

1. The client MUST use the Random ByteIO message “rbyteio:read”.
2. The client MUST use “0” as value for rbyteio:read/rbyteio:start-offset
3. The client MUST use “12” as value for rbyteio:read/rbyteio:bytes-per-block
4. The client MUST use “2” as value for rbyteio:read/rbyteio:num-blocks
5. The client MUST use “6” as value for rbyteio:read/rbyteio:stride
6. The client MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:read/rbyteio:transfer-information/@transfer-mechanism
7. The server MUST use the Random ByteIO message “rbyteio:readResponse”.
8. The server MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:readResponse/rbyteio:transfer-information/@transfer-mechanism
9. The server MUST return “MDEwMjAzMDQwNTA2MDQwNTA2MDcwODA5” as value for rbyteio:readResponse/rbyteio:transfer-information

4.5 rbyteio:write operation (i)

The Random ByteIO interoperability client invokes the method “rbyteio:write” to indicate that she wishes to write a set of bytes into the addressed resource.

In this case, the request describes a consecutive sequence of 6 bytes, starting at and including the byte at position 20. Interpreted as a string in US-ASCII, this block of bytes equivalents to “++++++”.

This request intends to test the general interoperability of the rbyteio:write operation.

To ensure correct behaviour on the server side the Random ByteIO client invokes the read operation on the resource so that it effectively retrieves the entire contents of the resource after the write operation completes.

4.5.1 Invoking the operation rbyteio:write()

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<rbyteio:write>
   <rbyteio:start-offset> 20 </rbyteio:start-offset>
   <rbyteio:bytes-per-block> 6 </rbyteio:bytes-per-block>
   <rbyteio:stride> 0 </rbyteio:stride>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”>
      KysrKys=
   </rbyteio:transfer-information>
</rbyteio:write>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<rbyteio:writeResponse>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”/>
</rbyteio:writeResponse>
```

4.5.2 Invoking the operation rbyteio:read()

To ensure correct processing on the server side, the client invokes rbyteio:read() similar to section 4.2 using the following rbyteio:read parameter values;

- start-offset = 0
- bytes-per-block = 60
- num-blocks = 1
- stride = 0

The result of this operation, i.e. the contents of the element //rbyteio:readResponse/rbyteio:transfer-information is part of the normative interoperability requirements (see below).

4.5.3 Interoperability requirements

1. The client MUST use the Random ByteIO message “rbyteio:write”.
2. The client MUST use “20” as value for rbyteio:write/rbyteio:start-offset
3. The client MUST use “5” as value for rbyteio:write/rbyteio:bytes-per-block
4. The client MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:write/rbyteio:transfer-information/@transfer-mechanism
5. The client MUST use “KysrKys=” as value for rbyteio:write/rbyteio:transfer-information
6. The server MUST use the Random ByteIO message “rbyteio:writeResponse”.
7. The server MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:writeResponse/rbyteio:transfer-information/@transfer-mechanism

8. Executing the XPath 1.0 operation “data(//rbyteio:readResponse/rbyteio:transfer-information)” MUST return the following value:
“0x30 0x31 0x30 0x32 0x30 0x33 0x30 0x34 0x30 0x35 0x30 0x36 0x30 0x37 0x30 0x38 0x30 0x39 0x31 0x30 0x2B 0x2B 0x2B 0x2B 0x2B 0x2B 0x31 0x34 0x31 0x35 0x31 0x36 0x31 0x37 0x31 0x38 0x31 0x39 0x32 0x30 0x32 0x31 0x32 0x32 0x32 0x33 0x32 0x34 0x32 0x35 0x32 0x36 0x32 0x37 0x32 0x38 0x32 0x39 0x33 0x30”, which, interpreted as a US-ASCII String, equivalents
“0102030405060708091 0++++++1415161718192021222324252627282930”.

4.6 rbyteio:write operation (ii)

The Random ByteIO interoperability client invokes the method “rbyteio:write” to indicate that she wishes to write a set of bytes into the addressed resource.

In this case, the request describes five sequences of two bytes each. These five blocks have a distance of 4 bytes between their respective start positions. The first block starts at and includes the byte at position 22. Interpreted as a string in US-ASCII, the byte array described in this request equivalents to “++++++++++”.

This request intends to test correct interpretation of the rbyteio:stride parameter of the rbyteio:write operation, and how the passed in new contents is constructed in case the value for rbyteio:stride exceeds the block size. It also tests repetitive block writing into the resource.

To ensure correct behaviour on the server side the Random ByteIO client invokes the read operation on the resource so that it effectively retrieves the entire contents of the resource after the write operation completes.

4.6.1 Invoking the operation rbyteio:write()

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<rbyteio:write>
   <rbyteio:start-offset> 22 </rbyteio:start-offset>
   <rbyteio:bytes-per-block> 2 </rbyteio:bytes-per-block>
   <rbyteio:stride> 4 </rbyteio:stride>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”>
      KysrKysrKysrKw==
   </rbyteio:transfer-information>
</rbyteio:write>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<rbyteio:writeResponse>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”/>
</rbyteio:writeResponse>
```

4.6.2 Invoking the operation rbyteio:read()

To ensure correct processing on the server side, the client invokes rbyteio:read() similar to section 4.2 using the following rbyteio:read parameter values;

- start-offset = 0
- bytes-per-block = 60
- num-blocks = 1
- stride = 0

The result of this operation, i.e. the contents of the element //rbyteio:readResponse/rbyteio:transfer-information is part of the normative interoperability requirements (see below).

4.6.3 Interoperability requirements

1. The client MUST use the Random ByteIO message “rbyteio:write”.
2. The client MUST use “22” as value for rbyteio:write/rbyteio:start-offset
3. The client MUST use “2” as value for rbyteio:write/rbyteio:bytes-per-block
4. The client MUST use “4” as value for rbyteio:write/rbyteio:stride
5. The client MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:write/rbyteio:transfer-information/@transfer-mechanism
6. The client MUST use “KysrKysrKysrKw==” as value for rbytei:write/rbyteio:transfer-information
7. The server MUST use the Random ByteIO message “rbyteio:writeResponse”.
8. The server MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:writeResponse/rbyteio:transfer-information/@transfer-mechanism
9. Executing the XPath 1.0 operation “data(//rbyteio:readResponse/rbyteio:transfer-information)” MUST return the following value:
   “0x30 0x31 0x30 0x32 0x30 0x33 0x30 0x34 0x30 0x35 0x30 0x36 0x30 0x37 0x30 0x38 0x30 0x39 0x31 0x30 0x31 0x31 0x2B 0x2B 0x31 0x33 0x2B 0x2B 0x31 0x35 0x2B 0x2B 0x31 0x37 0x2B 0x2B 0x31 0x39 0x2B 0x2B 0x32 0x31 0x32 0x32 0x32 0x33 0x32 0x34 0x32 0x35 0x32 0x36 0x32 0x37 0x32 0x38 0x32 0x39 0x33 0x30”, which, interpreted as a US-ASCII String, equivalents
   “01020304050607080910 11++13++15++17++19++21222324252627282930”.

4.7 rbyteio:write operation (iii)

The Random ByteIO interoperability client invokes the method “rbyteio:write” to indicate that she wishes to write a set of bytes into the addressed resource.

In this case, the request describes two sequences of 6 bytes each. These two blocks have a distance of 3 bytes between their respective start positions. The first block starts at and includes the byte at position 0. Interpreted as a string in US-ASCII, the byte array described in this request equivalents to “??????++++++”.

This request intends to test correct interpretation of the rbyteio:stride parameter of the rbyteio:write operation, and how the passed in new contents is constructed in case the value for rbyteio:stride is smaller than the block size. It also tests repetitive block writing into the resource.

To ensure correct behaviour on the server side the Random ByteIO client invokes the read operation on the resource so that it effectively retrieves the entire contents of the resource after the write operation completes.

4.7.1 Invoking the operation rbyteio:write()

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<rbyteio:write>
   <rbyteio:start-offset> 0 </rbyteio:start-offset>
   <rbyteio:bytes-per-block> 6 </rbyteio:bytes-per-block>
   <rbyteio:stride> 3 </rbyteio:stride>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”>
      Pz8/Pz8/KysrKysr
   </rbyteio:transfer-information>
</rbyteio:write>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<rbyteio:writeResponse>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”/>
</rbyteio:writeResponse>
```

4.7.2 Invoking the operation rbyteio:read()

To ensure correct processing on the server side, the client invokes rbyteio:read() similar to section 4.2 using the following rbyteio:read parameter values;

- start-offset = 0
- bytes-per-block = 60
- num-blocks = 1
- stride = 0

The result of this operation, i.e. the contents of the element //rbyteio:readResponse/rbyteio:transfer-information is part of the normative interoperability requirements (see below).

4.7.3 Interoperability requirements

1. The client MUST use the Random ByteIO message “rbyteio:write”.
2. The client MUST use “0” as value for rbyteio:write/rbyteio:start-offset
3. The client MUST use “6” as value for rbyteio:write/rbyteio:bytes-per-block
4. The client MUST use “3” as value for rbyteio:write/rbyteio:stride
5. The client MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:write/rbyteio:transfer-information/@transfer-mechanism
6. The client MUST use “Pz8/Pz8/KysrKysr” as value for rbytei:write/rbyteio:transfer-information
7. The server MUST use the Random ByteIO message “rbyteio:writeResponse”.
8. The server MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:writeResponse/rbyteio:transfer-information/@transfer-mechanism
9. Executing the XPath 1.0 operation “data(//rbyteio:readResponse/rbyteio:transfer-information)” MUST return the following value:
   “0x3F 0x3F 0x3F 0x2B 0x2B 0x2B 0x2B 0x2B 0x2B 0x35 0x30 0x36 0x30 0x37 0x30 0x38 0x30 0x39 0x31 0x30 0x31 0x31 0x31 0x32 0x31 0x33 0x31 0x34 0x31 0x35 0x31 0x36 0x31 0x37 0x31 0x38 0x31 0x39 0x32 0x30 0x32 0x31 0x32 0x32 0x32 0x33 0x32 0x34 0x32 0x35 0x32 0x36 0x32 0x37 0x32 0x38 0x32 0x39 0x33 0x30”, which, interpreted as a US-ASCII String, equivalents
   “???++++++5060708091011121314151617181920212223242526272829 30”.

4.8 rbyteio:append operation

The Random ByteIO interoperability client invokes the method “rbyteio:append” to indicate that she wishes to append a set of bytes to the addressed resource.

In this case, the request describes a consecutive sequence of 6 bytes. Interpreted as a string in US-ASCII, this block of bytes equivalents to “++++++”.

To ensure correct behaviour on the server side the Random ByteIO client invokes the read operation on the resource so that it effectively retrieves the entire contents of the resource after the write operation completes.

4.8.1 Invoking the operation rbyteio:write()

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<rbyteio:append>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”>
      KysrKysr
   </rbyteio:transfer-information>
</rbyteio:append>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<rbyteio:appendResponse>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”/>
</rbyteio:appendResponse>
```

4.8.2 Invoking the operation rbyteio:read()

To ensure correct processing on the server side, the client invokes rbyteio:read() similar to section 4.2 using the following rbyteio:read parameter values;

- start-offset = 0
- bytes-per-block = 66
- num-blocks = 1
- stride = 0

The result of this operation, i.e. the contents of the element //rbyteio:readResponse/rbyteio:transfer-information is part of the normative interoperability requirements (see below).

4.8.3 Interoperability requirements

1. The client MUST use the Random ByteIO message “rbyteio:append”.
2. The client MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:append/rbyteio:transfer-information/@transfer-mechanism
3. The client MUST use “KysrKysr” as value for rbyteio:append/rbyteio:transfer-information
4. The server MUST use the Random ByteIO message “rbyteio:appendResponse”.

5. The server MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:appendResponse/rbyteio:transfer-information/@transfer-mechanism
6. Executing the XPath 1.0 operation “data(//rbyteio:readResponse/rbyteio:transfer-information)” MUST return the following value:
“0x30 0x31 0x30 0x32 0x30 0x33 0x30 0x34 0x30 0x35 0x30 0x36 0x30 0x37 0x30 0x38 0x30 0x39 0x31 0x30 0x31 0x31 0x31 0x32 0x31 0x33 0x31 0x34 0x31 0x35 0x31 0x36 0x31 0x37 0x31 0x38 0x31 0x39 0x32 0x30 0x32 0x31 0x32 0x32 0x32 0x33 0x32 0x34 0x32 0x35 0x32 0x36 0x32 0x37 0x32 0x38 0x32 0x39 0x33 0x30 0x2B 0x2B 0x2B 0x2B 0x2B 0x2B”, which, interpreted as a US-ASCII String, equivalents “0102030405060708091011121314151617181920212223242526272829 30++++++”.

4.9 rbyteio:truncAppend operation

The Random ByteIO interoperability client invokes the method “rbyteio:truncAppend” to indicate that she wishes to append a set of bytes to the addressed resource right after the resource has been truncated to a specified length.

In this case, the request describes a consecutive sequence of 6 bytes. Interpreted as a string in US-ASCII, this block of bytes equivalents to “++++++”. The length of the resource before appending is set to 30 bytes.

To ensure correct behaviour on the server side the Random ByteIO client invokes the read operation on the resource so that it effectively retrieves the entire contents of the resource after the write operation completes.

4.9.1 Invoking the operation rbyteio:write()

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<rbyteio:truncAppend>
   <rbyteio:offset> 30 </rbyteio:offset>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”>
      KysrKysr
   </rbyteio:transfer-information>
</rbyteio:truncAppend>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<rbyteio:truncAppendResponse>
   <rbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”/>
</rbyteio:truncAppendResponse>
```

4.9.2 Invoking the operation rbyteio:read()

To ensure correct processing on the server side, the client invokes rbyteio:read() similar to section 4.2 using the following rbyteio:read parameter values;

- start-offset = 0
- bytes-per-block = 36
- num-blocks = 1

- stride = 0

The result of this operation, i.e. the contents of the element //rbyteio:readResponse/rbyteio:transfer-information is part of the normative interoperability requirements (see below).

4.9.3 Interoperability requirements

1. The client MUST use the Random ByteIO message “rbyteio:truncAppend”.
2. The client MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:truncAppend/rbyteio:transfer-information/@transfer-mechanism
3. The client MUST use “KysrKysr” as value for rbyteio:truncAppend/rbyteio:transfer-information
4. The server MUST use the Random ByteIO message “rbyteio:truncAppendResponse”.
5. The server MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for rbyteio:truncAppendResponse/rbyteio:transfer-information/@transfer-mechanism
6. Executing the XPath 1.0 operation “data(//rbyteio:readResponse/rbyteio:transfer-information)” MUST return the following value:
“0x30 0x31 0x30 0x32 0x30 0x33 0x30 0x34 0x30 0x35 0x30 0x36 0x30 0x37 0x30 0x38 0x30 0x39 0x31 0x30 0x31 0x31 0x31 0x32 0x31 0x33 0x31 0x34 0x31 0x35 0x2B 0x2B 0x2B 0x2B 0x2B 0x2B”, which, interpreted as a US-ASCII String, equivalents “0102030405060708091011121314 15++++++”.

5. Streamable ByteIO Interoperability Tests

This chapter defines the Interoperability tests for the Streamable ByteIO interface specification. The tests are defined in a logical order. However, this order of definition does not imply any particular order of execution.

5.1 wsrf-rp:GetResourceProperty operation

The Streamable ByteIO interoperability client invokes the method “wsrf-rp:GetResourceProperty” to indicate that she wishes to get the value of a given resource property.

In this case, the resource property is “sbyteio:Readable”.

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<wsrf-rp:GetResourceProperty> sbyteio:Readable </wsrf-rp:GetResourceProperty>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<wsrf-rp:GetResourcePropertyResponse>
   <sbyteio:Readable> true </sbyteio:Readable>
</wsrf-rp:GetResourcePropertyResponse>
```

5.1.1 Interoperability requirements

1. The client MUST use the WSRF RP message “wsrf-rp:GetResourceProperty”

2. The client MUST use the QName “sbyteio:Readable” to request the Streamable ByteIO Readable Resource Property.
3. The server MUST use the WSRF RP message “wsrf-rp:GetResourcePropertyResponse”.
4. The server MUST return the value “true” for the Resource Property “sbyteio:Readable”.

5.2 sbyteio:seekRead operation

The Streamable ByteIO interoperability client invokes the method “sbyteio:seekRead” to indicate that she wishes to read a set of bytes from the addressed resource.

In this case, the request describes a consecutive sequence of 12 bytes starting at and including the byte at position 20. Interpreted as a string in US-ASCII, this block of bytes equivalents to “111213141516”.

This request intends to test the general interoperability of the sbyteio:seekRead operation.

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<sbyteio:seekRead>
   <sbyteio:offset> 20 </sbyteio:offset>
   <sbyteio:seek-origin>
      http://schemas.ggf.org/byteio/2005/10/streamable-access/seek-origins/beginning
   </sbyteio:seek-origin>
   <sbyteio:num-bytes> 12 </sbyteio:num-bytes>
   <sbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”/>
</sbyteio:read>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<sbyteio:seekReadResponse>
   <sbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”>
      MTExMjEzMTQxNTE2
   </sbyteio:transfer-information>
</sbyteio:seekReadResponse>
```

5.2.1 Interoperability requirements

1. The client MUST use the Streamable ByteIO message “sbyteio:seekRead”.
2. The client MUST use “20” as value for sbyteio:seekRead/sbyteio:offset
3. The client MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for sbyteio:seekRead/sbyteio:transfer-information/@transfer-mechanism
4. The client MUST use “12” as value for sbyteio:seekRead/sbyteio:nuum-bytes
5. The server MUST use the Streamable ByteIO message “sbyteio:seekReadResponse”.
6. The server MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for sbyteio:seekReadResponse/sbyteio:transfer-information/@transfer-mechanism
7. The server MUST return “MTExMjEzMTQxNTE2” as value for sbyteio:seekReadResponse/sbyteio:transfer-information

5.3 sbyteio:seekWrite operation

The Streamable ByteIO interoperability client invokes the method “sbyteio:seekWrite” to indicate that she wishes to write a set of bytes into the addressed resource.

In this case, the request describes a consecutive sequence of 6 bytes, starting at and including the byte at position 20. Interpreted as a string in US-ASCII, this block of bytes equivalents to “++++++”.

This request intends to test the general interoperability of the sbyteio:seekWrite operation.

To ensure correct behaviour on the server side the Streamable ByteIO client invokes the seekRead operation on the resource so that it effectively retrieves the entire contents of the resource after the write operation completes.

5.3.1 Invoking the operation sbyteio:seekWrite()

The following is a non-normative example of the element the ByteIO client adds to the SOAP body of the request message:

```
<sbyteio:seekWrite>
   <sbyteio:offset> 20 </sbyteio:offset>
   <sbyteio:seek-origin>
      http://schemas.ggf.org/byteio/2005/10/streamable-access/seek-origins/beginning
   </sbyteio:seek-origin>
   <sbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”>
      KysrKys=
   </sbyteio:transfer-information>
</sbyteio:seekWrite>
```

The following is a non-normative example of the element the ByteIO server adds to the SOAP body of the response message:

```
<sbyteio:seekWriteResponse>
   <sbyteio:transfer-information transfer-mechanism=
”http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple”/>
</sbyteio:seekWriteResponse>
```

5.3.2 Invoking the operation sbyteio:seekRead()

To ensure correct processing on the server side, the client invokes sbyteio:seekRead() similar to section 5.2 using the following sbyteio:seekRead parameter values;

- offset = 0
- seek-origin = http://schemas.ggf.org/byteio/2005/10/streamable-access/seek-origins/beginning
- num-bytes = 60

The result of this operation, i.e. the contents of the element //sbyteio:seekReadResponse/sbyteio:transfer-information is part of the normative interoperability requirements (see below).

5.3.3 Interoperability requirements

1. The client MUST use the Streamable ByteIO message “sbyteio:seekWrite”.
2. The client MUST use “20” as value for sbyteio:seekWrite/sbyteio:offset
3. The client MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for sbyteio:seekWrite/sbyteio:transfer-information/@transfer-mechanism
4. The client MUST use “KysrKys=” as value for sbyteio:seekWrite/sbyteio:transfer-information
5. The server MUST use the Streamable ByteIO message “sbyteio:seekWriteResponse”.
6. The server MUST use “http://schemas.ggf.org/byteio/2005/10/transfer-mechanisms/simple” as value for sbyteio:seekWriteResponse/sbyteio:transfer-information/@transfer-mechanism

Executing the XPath 1.0 operation “data(//sbyteio:seekReadResponse/sbyteio:transfer-information)” MUST return the following value:
“0x30 0x31 0x30 0x32 0x30 0x33 0x30 0x34 0x30 0x35 0x30 0x36 0x30 0x37 0x30 0x38 0x30 0x39 0x31 0x30 0x2B 0x2B 0x2B 0x2B 0x2B 0x2B 0x31 0x34 0x31 0x35 0x31 0x36 0x31 0x37 0x31 0x38 0x31 0x39 0x32 0x30 0x32 0x31 0x32 0x32 0x32 0x33 0x32 0x34 0x32 0x35 0x32 0x36 0x32 0x37 0x32 0x38 0x32 0x39 0x33 0x30”, which, interpreted as a US-ASCII String, equivalents “01020304050607080910++++++1415161718192021222324252627282930”.

Author Information

Editor:
Michel Drescher
Fujitsu Laboratories of Europe Ltd.
Hayes Park Central, Hayes End Road
Hayes, Middlesex UB4 8FE
United Kingdom
E-mail: Michel.Drescher@uk.fujitsu.com

Acknowledgements

The document authors wish to express their sincere gratitude to everybody who has contributed to this document, while apologising for anybody unintentionally omitted, in alphabetical order:

Malcolm Atkinson, Neil Chue Hong, Amy Krause, Mark Morgan.

Intellectual Property Statement

Full Copyright Notice

References

[GFD.1] C. Catlett, *Global Grid Forum Documents and Recommendations: Process and Requirements*, Global Grid Forum Document GFD.1, April 2002, http://www.gridforum.org/documents/GFD.1.pdf

[RFC2119] S.Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF TFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[OGSA BP 1.0] I. Foster, T. Maguire, D. Snelling, *OGSA WSRF Basic Profile 1.0, Global Grid Forum Document*, https://forge.gridforum.org/projects/ogsa-wg/document/draft-ggf-ogsa-wsrf-basic-profile/en/15, GWS-R (draft-ggf-ogsa-wsrf-basic-profile-021), 6 July 2005.

Appendix A Normative Interoperability Interface

This Appendix normatively defines the interface any ByteIO implementation must support either directly or indirectly:

```
<?xml version="1.0" encoding="utf-8"?>
<!--
   The GGF takes no position regarding the validity or scope of any intellectual property or
other rights that might be claimed to pertain to the implementation or use of the technology
described in this document or the extent to which any license under such rights might or might
not be available; neither does it represent that it has made any effort to identify any such
rights. Copies of claims of rights made available for publication and any assurances of
licenses to be made available, or the result of an attempt made to obtain a general license or
permission for the use of such proprietary rights by implementers or users of this specification
can be obtained from the GGF Secretariat.

   The GGF invites any interested party to bring to its attention any copyrights, patents or
patent applications, or other proprietary rights which may cover technology that may be
required to practice this recommendation. Please address the information to the GGF
Executive Director.

   Copyright (C) Open Grid Forum (2006). All Rights Reserved.

   This document and translations of it may be copied and furnished to others, and derivative
works that comment on or otherwise explain it or assist in its implementation may be
prepared, copied, published and distributed, in whole or in part, without restriction of any kind,
provided that the above copyright notice and this paragraph are included on all such copies
and derivative works. However, this document itself may not be modified in any way, such as
by removing the copyright notice or references to the GGF or other organizations, except as
needed for the purpose of developing Grid Recommendations in which case the procedures
for copyrights defined in the GGF Document process must be followed, or as required to
translate it into languages other than English.

   The limited permissions granted above are perpetual and will not be revoked by the GGF or
its successors or assigns.
   This document and the information contained herein is provided on an "AS IS" basis and
THE GLOBAL GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

-->
<wsdl:definitions name="ByteIOInterop-Service"
   targetNamespace="http://schemas.ggf.org/byteio/2006/07/interop"
   xmlns:interop="http://schemas.ggf.org/byteio/2006/07/interop"
   xmlns:wsa="http://www.w3.org/2005/08/addressing"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <!-- ByteIOInterop::Types -->
   <wsdl:types>
      <xsd:schema targetNamespace="http://schemas.ggf.org/byteio/2006/07/interop"
                  elementFormDefault="qualified" attributeFormDefault="unqualified">
         <xsd:import namespace="http://www.w3.org/2005/08/addressing"
```

```
                    schemaLocation="http://www.w3.org/2005/08/addressing/ws-addr.xsd"/>
      <xsd:element name="createResource" nillable="true"/>
      <xsd:element name="createResourceResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="wsa:EndpointReference"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="deleteResource">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="wsa:EndpointReference"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>

  <!-- ByteIOInterop::Messages -->
  <wsdl:message name="createResource">
    <wsdl:part name="createResource" element="interop:createResource"/>
  </wsdl:message>

  <wsdl:message name="createResourceResponse">
    <wsdl:part name="createResourceResponse"
               element="interop:createResourceResponse"/>
  </wsdl:message>

  <wsdl:message name="deleteResource">
    <wsdl:part name="deleteResource" element="interop:deleteResource"/>
  </wsdl:message>

  <!-- ByteIOInterop::PortType -->
  <wsdl:portType name="ByteIOInterop">

    <wsdl:operation name="CreateResource">
      <wsdl:input message="interop:createResource"/>
      <wsdl:output message="interop:createResourceResponse"/>
    </wsdl:operation>

    <wsdl:operation name="DeleteResource">
      <wsdl:input message="interop:deleteResource"/>
    </wsdl:operation>

  </wsdl:portType>

  <!-- ByteIOInterop::Binding -->
  <wsdl:binding name="ByteIOInterop-Binding" type="interop:ByteIOInterop">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsaw:UsingAddressing wsdl:required="true"/>

    <wsdl:operation name="CreateResource">
      <soap:operation soapAction="http://schemas.ggf.org/byteio/2006/07/interop/create"/>
      <wsdl:input>
        <soap:body use="literal"
```

```
                        namespace="http://schemas.ggf.org/byteio/2006/07/interop"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal"
                        namespace="http://schemas.ggf.org/byteio/2006/07/interop"/>
        </wsdl:output>
      </wsdl:operation>

      <wsdl:operation name="DeleteResource">
        <soap:operation soapAction="http://schemas.ggf.org/byteio/2006/07/interop/delete"/>
        <wsdl:input>
          <soap:body use="literal"
                       namespace="http://schemas.ggf.org/byteio/2006/07/interop"/>
        </wsdl:input>
      </wsdl:operation>

    </wsdl:binding>

    <!-- ByteIOInterop::Service -->
    <wsdl:service name="ByteIOInterop-Service">
      <wsdl:port name="ByteIOInterop-Service" binding="interop:ByteIOInterop-Binding">
        <soap:address location=""/>
      </wsdl:port>
    </wsdl:service>

</wsdl:definitions>
```

Note: The node “//wsdl:definitions/wsdl:service/wsdl:port/soap:address@location” MAY be altered to suit the needs of concrete interoperability tests without violating the normative requirements of this document.

Appendix B Normative RandomByteIO Binding for Interoperability Tests

This section normatively defines the transport binding any Random Access ByteIO implementation MUST support to participate in Interoperability tests. This does not restrict the implementation by no means to support other transport bindings.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
   The GGF takes no position regarding the validity or scope of any intellectual property or
other rights that might be claimed to pertain to the implementation or use of the technology
described in this document or the extent to which any license under such rights might or might
not be available; neither does it represent that it has made any effort to identify any such
rights. Copies of claims of rights made available for publication and any assurances of
licenses to be made available, or the result of an attempt made to obtain a general license or
permission for the use of such proprietary rights by implementers or users of this specification
can be obtained from the GGF Secretariat.

   The GGF invites any interested party to bring to its attention any copyrights, patents or
patent applications, or other proprietary rights which may cover technology that may be
required to practice this recommendation. Please address the information to the GGF
Executive Director.

   Copyright (C) Open Grid Forum (2006). All Rights Reserved.

   This document and translations of it may be copied and furnished to others, and derivative
works that comment on or otherwise explain it or assist in its implementation may be
prepared, copied, published and distributed, in whole or in part, without restriction of any kind,
provided that the above copyright notice and this paragraph are included on all such copies
and derivative works. However, this document itself may not be modified in any way, such as
by removing the copyright notice or references to the GGF or other organizations, except as
needed for the purpose of developing Grid Recommendations in which case the procedures
for copyrights defined in the GGF Document process must be followed, or as required to
translate it into languages other than English.

   The limited permissions granted above are perpetual and will not be revoked by the GGF or
its successors or assigns.
   This document and the information contained herein is provided on an "AS IS" basis and
THE GLOBAL GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

-->

<wsdl:definitions name="RandomByteIOBinding"
   targetNamespace="http://schemas.ggf.org/byteio/2005/10/random-access"
   xmlns:byteio="http://schemas.ggf.org/byteio/2005/10/byte-io"
   xmlns:rbyteio="http://schemas.ggf.org/byteio/2005/10/random-access"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl">

   <wsdl:import namespace="http://schemas.ggf.org/byteio/2005/10/random-access"
                location="./rbyteio-porttype.wsdl"/>

   <wsdl:binding name="RandomByteIOBinding" type="rbyteio:RandomByteIO">
```

```
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
      <wsaw:UsingAddressing wsdl:required="true"/>
      <!-- ===== WSRF operations ===== -->
      <wsdl:operation name="GetResourceProperty">
        <soap:operation soapAction="http://docs.oasis-open.org/wsrf/rpw-
2/GetResourceProperty/GetResourcePropertyRequest"/>
        <wsdl:input>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:output>
        <wsdl:fault name="InvalidResourcePropertyQNameFault">
          <soap:fault name="InvalidResourcePropertyQNameFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="GetMultipleResourceProperties">
        <soap:operation soapAction="http://docs.oasis-open.org/wsrf/rpw-
2/GetMultipleResourceProperties/GetMultipleResourcePropertiesRequest"/>
        <wsdl:input>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:output>
        <wsdl:fault name="InvalidResourcePropertyQNameFault">
          <soap:fault name="InvalidResourcePropertyQNameFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="QueryResourceProperties">
        <soap:operation soapAction="http://docs.oasis-open.org/wsrf/rpw-
2/QueryResourceProperties/QueryResourcePropertiesRequest"/>
        <wsdl:input>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:output>
        <wsdl:fault name="InvalidResourcePropertyQNameFault">
          <soap:fault name="InvalidResourcePropertyQNameFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnknownQueryExpressionDialectFault">
          <soap:fault name="UnknownQueryExpressionDialectFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InvalidQueryExpressionFault">
          <soap:fault name="InvalidQueryExpressionFault" use="literal"/>
        </wsdl:fault>
```

```
        <wsdl:fault name="QueryEvaluationErrorFault">
          <soap:fault name="QueryEvaluationErrorFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <!-- ===== ByteIO::Random-Access ===== -->
      <wsdl:operation name="read">
        <soap:operation soapAction="http://schemas.ggf.org/byteio/2005/10/random-
access/read"/>
        <wsdl:input>
          <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/random-access"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/random-access"/>
        </wsdl:output>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedTransferFault">
          <soap:fault name="UnsupportedTransferFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ReadNotPermittedFault">
          <soap:fault name="ReadNotPermittedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="CustomFault">
          <soap:fault name="CustomFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="write">
        <soap:operation soapAction="http://schemas.ggf.org/byteio/2005/10/random-
access/write"/>
        <wsdl:input>
          <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/random-access"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/random-access"/>
        </wsdl:output>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedTransferFault">
          <soap:fault name="UnsupportedTransferFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="WriteNotPermittedFault">
          <soap:fault name="WriteNotPermittedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="CustomFault">
          <soap:fault name="CustomFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="append">
        <soap:operation soapAction="http://schemas.ggf.org/byteio/2005/10/random-
access/append"/>
```

```
        <wsdl:input>
          <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/random-access"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/random-access"/>
        </wsdl:output>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedTransferFault">
          <soap:fault name="UnsupportedTransferFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="WriteNotPermittedFault">
          <soap:fault name="WriteNotPermittedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="CustomFault">
          <soap:fault name="CustomFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="truncAppend">
        <soap:operation soapAction="http://schemas.ggf.org/byteio/2005/10/random-
access/truncAppend"/>
        <wsdl:input>
          <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/random-access"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/random-access"/>
        </wsdl:output>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedTransferFault">
          <soap:fault name="UnsupportedTransferFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="TruncateNotPermittedFault">
          <soap:fault name="TruncateNotPermittedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="WriteNotPermittedFault">
          <soap:fault name="WriteNotPermittedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="CustomFault">
          <soap:fault name="CustomFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="RandomByteIOService">
      <wsdl:port name="RandomByteIO" binding="rbyteio:RandomByteIOBinding">
        <soap:address location="http://www.example.org/service/rbyteio"/>
      </wsdl:port>
    </wsdl:service>
```

```
</wsdl:definitions>
```

Note: The node “//wsdl:definitions/wsdl:service/wsdl:port/soap:address@location” MAY be altered to suit the needs of concrete interoperability tests without violating the normative requirements of this document.

Appendix C Normative StreamableByteIO Binding for Interoperability Tests

This section normatively defines the transport binding any Streamable Access ByteIO implementation MUST support to participate in Interoperability tests. This does not restrict the implementation by no means to support other transport bindings.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
   The GGF takes no position regarding the validity or scope of any intellectual property or
other rights that might be claimed to pertain to the implementation or use of the technology
described in this document or the extent to which any license under such rights might or might
not be available; neither does it represent that it has made any effort to identify any such
rights. Copies of claims of rights made available for publication and any assurances of
licenses to be made available, or the result of an attempt made to obtain a general license or
permission for the use of such proprietary rights by implementers or users of this specification
can be obtained from the GGF Secretariat.

   The GGF invites any interested party to bring to its attention any copyrights, patents or
patent applications, or other proprietary rights which may cover technology that may be
required to practice this recommendation. Please address the information to the GGF
Executive Director.

   Copyright (C) Open Grid Forum (2006). All Rights Reserved.

   This document and translations of it may be copied and furnished to others, and derivative
works that comment on or otherwise explain it or assist in its implementation may be
prepared, copied, published and distributed, in whole or in part, without restriction of any kind,
provided that the above copyright notice and this paragraph are included on all such copies
and derivative works. However, this document itself may not be modified in any way, such as
by removing the copyright notice or references to the GGF or other organizations, except as
needed for the purpose of developing Grid Recommendations in which case the procedures
for copyrights defined in the GGF Document process must be followed, or as required to
translate it into languages other than English.

   The limited permissions granted above are perpetual and will not be revoked by the GGF or
its successors or assigns.
   This document and the information contained herein is provided on an "AS IS" basis and
THE GLOBAL GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE
INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED
WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

-->
<wsdl:definitions name="StreamableByteIOBinding"
   targetNamespace="http://schemas.ggf.org/byteio/2005/10/streamable-access"
   xmlns:byteio="http://schemas.ggf.org/byteio/2005/10/byte-io"
   xmlns:sbyteio="http://schemas.ggf.org/byteio/2005/10/streamable-access"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:wsaw="http://www.w3.org/2006/02/addressing/wsdl">

   <wsdl:import namespace="http://schemas.ggf.org/byteio/2005/10/streamable-access"
                location="./sbyteio-porttype.wsdl"/>

   <wsdl:binding name="RandomByteIOBinding" type="sbyteio:StreamableByteIO">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
```

```
      <wsaw:UsingAddressing wsdl:required="true"/>
      <!-- ===== WSRF operations ===== -->
      <wsdl:operation name="GetResourceProperty">
        <soap:operation soapAction="http://docs.oasis-open.org/wsrf/rpw-
2/GetResourceProperty/GetResourcePropertyRequest"/>
        <wsdl:input>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:output>
        <wsdl:fault name="InvalidResourcePropertyQNameFault">
          <soap:fault name="InvalidResourcePropertyQNameFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="GetMultipleResourceProperties">
        <soap:operation soapAction="http://docs.oasis-open.org/wsrf/rpw-
2/GetMultipleResourceProperties/GetMultipleResourcePropertiesRequest"/>
        <wsdl:input>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:output>
        <wsdl:fault name="InvalidResourcePropertyQNameFault">
          <soap:fault name="InvalidResourcePropertyQNameFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="QueryResourceProperties">
        <soap:operation soapAction="http://docs.oasis-open.org/wsrf/rpw-
2/QueryResourceProperties/QueryResourcePropertiesRequest"/>
        <wsdl:input>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rpw-2"/>
        </wsdl:output>
        <wsdl:fault name="InvalidResourcePropertyQNameFault">
          <soap:fault name="InvalidResourcePropertyQNameFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnknownQueryExpressionDialectFault">
          <soap:fault name="UnknownQueryExpressionDialectFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="InvalidQueryExpressionFault">
          <soap:fault name="InvalidQueryExpressionFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="QueryEvaluationErrorFault">
```

```
            <soap:fault name="QueryEvaluationErrorFault" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="Destroy">
         <soap:operation soapAction="http://docs.oasis-open.org/wsrf/rlw-
2/Destroy/DestroyRequest"/>
         <wsdl:input>
            <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rlw-2"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rlw-2"/>
         </wsdl:output>
         <wsdl:fault name="ResourceUnknownFault">
            <soap:fault name="ResourceUnknownFault" use="literal"/>
         </wsdl:fault>
         <wsdl:fault name="ResourceNotDestroyedFault">
            <soap:fault name="ResourceNotDestroyedFault" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="SetTerminationTime">
         <soap:operation soapAction="http://docs.oasis-open.org/wsrf/rlw-
2/SetTerminationTime/SetTerminationTimeRequest"/>
         <wsdl:input>
            <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rlw-2"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal" namespace="http://docs.oasis-open.org/wsrf/rlw-2"/>
         </wsdl:output>
         <wsdl:fault name="ResourceUnknownFault">
            <soap:fault name="ResourceUnknownFault" use="literal"/>
         </wsdl:fault>
         <wsdl:fault name="UnableToSetTerminationTimeFault">
            <soap:fault name="UnableToSetTerminationTimeFault" use="literal"/>
         </wsdl:fault>
         <wsdl:fault name="TerminationTimeChangeRejectedFault">
            <soap:fault name="TerminationTimeChangeRejectedFault" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
      <!-- ===== ByteIO::Streamable-Access ===== -->
      <wsdl:operation name="seekRead">
         <soap:operation soapAction="http://schemas.ggf.org/byteio/2005/10/random-
access/seekRead"/>
         <wsdl:input>
            <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/streamable-access"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/streamable-access"/>
         </wsdl:output>
         <wsdl:fault name="ResourceUnknownFault">
            <soap:fault name="ResourceUnknownFault" use="literal"/>
         </wsdl:fault>
         <wsdl:fault name="UnsupportedTransferFault">
            <soap:fault name="UnsupportedTransferFault" use="literal"/>
         </wsdl:fault>
```

```
        <wsdl:fault name="SeekNotPermittedFault">
          <soap:fault name="SeekNotPermittedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="ReadNotPermittedFault">
          <soap:fault name="ReadNotPermittedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="CustomFault">
          <soap:fault name="CustomFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="seekWrite">
        <soap:operation soapAction="http://schemas.ggf.org/byteio/2005/10/random-
access/seekWrite"/>
        <wsdl:input>
          <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/streamable-access"/>
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal"
namespace="http://schemas.ggf.org/byteio/2005/10/streamable-access"/>
        </wsdl:output>
        <wsdl:fault name="ResourceUnknownFault">
          <soap:fault name="ResourceUnknownFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="UnsupportedTransferFault">
          <soap:fault name="UnsupportedTransferFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="WriteNotPermittedFault">
          <soap:fault name="WriteNotPermittedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="SeekNotPermittedFault">
          <soap:fault name="SeekNotPermittedFault" use="literal"/>
        </wsdl:fault>
        <wsdl:fault name="CustomFault">
          <soap:fault name="CustomFault" use="literal"/>
        </wsdl:fault>
      </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="StreamableByteIOService">
      <wsdl:port name="StreamableByteIO" binding="sbyteio:StreamableByteIOBinding">
        <soap:address location="http://www.example.org/service/sbyteio"/>
      </wsdl:port>
    </wsdl:service>

</wsdl:definitions>
```

Note: The node “//wsdl:definitions/wsdl:service/wsdl:port/soap:address@location” MAY be altered to suit the needs of concrete interoperability tests without violating the normative requirements of this document.