
Meet the specs: WS-RT 1.0 operations, Part 2

Take a closer look at how the WS-ResourceTransfer 1.0 specification extends the Create operation

Skill Level: Introductory

[Kane Scarlett \(kane@us.ibm.com\)](mailto:kane@us.ibm.com)
developerWorks Editor
IBM

06 Feb 2007

Meet the WS-ResourceTransfer 1.0 initial draft specification, a proposed open standard that extends certain operations by allowing fragments of XML code in a single resource to be addressed instead of having to affect the entire resource. This article provides a closer look at how the WS-ResourceTransfer 1.0 specification extends the `Create` operation.

This series of *Meet the specs* articles focuses on various components of the WS-ResourceTransfer 1.0 initial draft specification. This article includes excerpted and paraphrased text from the original specification. To view the specification in its entirety, see [Resources](#).

The WS-ResourceTransfer 1.0 initial draft specification (WS-RT) defines extensions to WS-Transfer, a general SOAP-based protocol for accessing XML representations of Web service-based resources. The family of Web services specifications, the "WS-" group, is designed to interoperate with other members of the family to deliver a set of tools for the Web services environment. As such, this specification relies on other WSs for such functions as message delivery and to express WS metadata.

WS-RT is an essential core component of a unified resource access protocol for Web services. The WS-RT extensions deal mostly with fragment-based access to resources to satisfy the common requirements of WS-ResourceFramework and WS-Management specifications. The specification goals are to fulfill these requirements:

- Define a standardized technique for accessing resources using semantics (`get`, `put`, `create`, and `delete`) familiar to system-management professionals
- Define WS-I BP 1.1-compliant WSDL 1.1 portTypes for the Web service methods described in the specification
- Define minimum requirements for compliance without constraining richer implementations
- Compose with other Web service specifications for secure, reliable, transacted message delivery
- Provide extensibility for more sophisticated or currently unanticipated scenarios
- Support a variety of encoding formats including SOAP 1.1 and SOAP 1.2 Envelopes

Let's start with WS-Transfer

WS-Transfer defines a mechanism for acquiring XML-based representations of entities using the Web service infrastructure. It defines two types of entities -- *resources* and *resource factories*. Resources are Web services that are addressable by an endpoint reference (as defined in WS-Addressing) that provides an XML representation. Resource factories are Web services that can create a new resource from an XML representation. What WS-T specifically defines is two operations for sending and receiving the representation of a given resource and two operations for creating and deleting a resource and its corresponding representation. WS-Transfer defines operations to the `get`, `put`, `create`, and `delete` representations of resources. WS-ResourceTransfer extends these operations to add to them the capability to operate on fragments of the resource representations.

In the first *Meet the specs* column on WS-RT, I provided a quick look at the four operations. The second article looked at the nuts-and-bolts level to see how WS-RT extends the `get` operation. Now let's dig a little deeper into how WS-RT extends the `create` operation. But first, a reminder on fragments and the QName and two XPath dialects.

Fragments, dialects, QName, and XPath

WS-ResourceTransfer extends these four operations so they can operate on fragments of the resource representations. So what's a fragment? A piece of XML code that is part of the resource. WS-RT imparts the power of *fragment access*, the ability to access a part of the resource's XML code.

Expression dialects form an expression that can be evaluated with respect to the XML document that represents the resource -- the de-referenced value of the expression is the part of the XML that is of interest. The expression can create a logical pointer to the fragment of XML that is of interest or it may craft a query that can be applied to the XML document to produce an evaluated result. (Remember: These expression dialects simply ID the appropriate fragment of the resource representation; it's the action that defines what happens to the fragment.)

Dialects include:

- **QName:** A simple dialect for expressions that uses a QName to reference the immediate children of the root element of the resource representation. QName is useful for simple resources with no XPath processing capability. The expression must evaluate to zero or more entire elements, each including the element name, any attributes, and its entire content. The QName dialect does not support computed values.

XPath is a sublanguage in an XSL style sheet that is used to identify XML elements for processing. It is also used to calculate numbers and manipulate strings. XPath and XSLT expressions are intertwined. For example, although XPath can reference a variable, the variable must be created and given a value in XSLT. XPath syntax is somewhat like the directory addressing in UNIX®, which uses a slash for the root directory as well as the separator between hierarchies.

Specification contributors

HP, IBM®, Intel, and Microsoft® collaborated to publish the WS-ResourceTransfer (WS-RT), an initial draft that describes an extension to the WS-Transfer specification. WS-Transfer defines standard messages for controlling resources using the familiar paradigms of *get*, *put*, *create*, and *delete*. The extensions deal primarily with fragment-based access to resources to satisfy the common requirements of WS-ResourceFramework and WS-Management.

- **XPath Level 1:** A dialect that uses an XPath to reference specific fragments of the resource; it is logically applied to the XML representation of the resource and the resulting node set is the resource fragment that is the subject of the message containing the expression. This dialect is useful for resources with limited XPath processing capability that do not need to support returning values computed from their resource representation. It can define references to any element, attribute, or value in the resource representation; expressions in this dialect must not evaluate to more than a single node. The dialect does not support computed values, and text and attribute nodes must be serialized using

the same serialization as for the XPath 1.0 dialect.

- XPath 1.0: This dialect differs from XPath Level 1 in that it supports a wider set of XPath function libraries. This dialect works well for any resource with full XPath processing capability or any resource that needs to support returning values computed from its resource representation. It can define references to any element, attribute, or value in the resource representation and can also be used to compute values from the resource representation. Implementations that support the full XPath 1.0 dialect must also support the XPath Level 1 dialect.

Now let's move on to the `create` operation.

Extending the create operation

In WS-Transfer, the `create` operation is used for creating a resource using an initial representation. The resource factory that receives a `create` request allocates a new resource that is initialized from the presented representation. The new resource is assigned a factory-service-determined endpoint reference that is returned in the response message.

Sometimes, the information required to create a resource may markedly differ from the initial representation (the value as realized by a subsequent `get` operation) and supplying the initial representation is not viable. WS-ResourceTransfer extends the `create` operation to create a resource from *zero or more* specified fragments of the XML representation. WS-ResourceTransfer also extends the operation so that any resource metadata can be created as part of the creation of the resource.

Listing 1 shows the extended outline for the `create` operation:

Listing 1. Extended outline for create

```
[Headers]
<wsrt:ResourceTransfer s:mustUnderstand="true"/>
[Action]
http://schemas.xmlsoap.org/ws/2004/09/transfer/Create
[Body]
<wsrt:Create Dialect="xs:anyURI"?>
  <wsmex:Metadata>resource metadata</wsmex:Metadata> ?
  <wsrt:Fragment>
    <wsrt:Expression>xs:any</wsrt:Expression> ?
    <wsrt:Value ...>xs:any</wsrt:Value>
  </wsrt:Fragment> *
</wsrt:Create>
```

These additional elements are applied as constraints on the outline in Listing 1:

- If `[Header]/wsrt:ResourceTransfer` is present and understood, a resource must process the `[Body]` in its entirety and comply with its content. If it is not present, a resource has to treat this request as described in WS-Transfer `create`.
- `[Body]/wsrt:Create` is an element that specifies the fragments of the resource representation to be initialized during resource creation; optionally, it specifies any resource metadata that is to be created as part of the creation of the resource. This element must be present if the `wsrt:ResourceTransfer` header is present.
- The `[Body]/wsrt:Create/@Dialect` URI indicates which expression dialect will be used to identify the fragment or fragments of the resource representation to be initialized during resource creation; this attribute has to be present when the message contains a `wsrt:Expression` element.
- When the optional element `[Body]/wsrt:Create/wsmex:Metadata` is present, it must contain at least one `wsmex:MetadataSection`. This is resource metadata to be created and initialized during the creation of the resource.

Note: A resource factory must generate an `InvalidMetadataFault` if the `create` request message contains a `wsmex:Dialect` that is not supported or if the resource metadata contains values that are not supported for the resource. The `[Body]/wsrt:Create/wsmex:Metadata` element may contain a `wsmex:MetadataSection` with a `wsmex:Dialect` of `http://schemas.xmlsoap.org/ws/2006/08/resourceTransfer`, allowing the requestor to specify the desired metadata as defined in this specification (such as life cycle metadata).

- The `[Body]/wsrt:Create/Fragment` element encompasses a single resource fragment to be initialized during the resource creation. If there are multiple fragment elements, the resource has to appear to have been created as though each fragment were processed in the sequence specified in the `create` message. If the request contains more fragment elements than the resource supports, the resource returns a fault, which should be `wsrt:MultipartLimitExceededFault`.
- When the optional `[Body]/wsrt:Create/Fragment/Expression` element is present, it contains an expression that identifies a resource fragment to be initialized during resource creation. The expression identifies the fragment in the resource representation as it appears after successful processing of the current fragment. If this element is absent, then that's like an expression that identifies the entire resource

representation.

- The `[Body]/wsrt:Create/Fragment/Value` element contains the data to be written to the resource representation; if the resource factory is unable to write the requested fragment, then it must generate a `CreateFault`.

If the resource factory accepts a `create` request, it then needs to reply with a response like that shown in Listing 2:

Listing 2. Reply when resource factory accepts create request

```
[Headers]
<wsrt:ResourceTransfer/>
[Action]
http://schemas.xmlsoap.org/ws/2004/09/transfer/CreateResponse
[Body]
<wxf:ResourceCreated>
  wsa:EndpointReferenceType
</wxf:ResourceCreated>
```

The additional constraints on the outline in Listing 2 are:

- A `[Headers]/wsrt:ResourceTransfer` header indicates that the response contains body content defined in WS1001 ResourceTransfer.
- The `[Body]/wxf:ResourceCreated` element contains the endpoint reference for the resource that was created. Any and all subsequent access to the resource must be achieved using this EPR.

If the request body contained a `wsrt:Create` element, then the new representation needs to be omitted in the response; otherwise the response must be as described in WS-Transfer.

An example `create` message using the QName dialect is shown in Listing 3 (only the message body is shown):

Listing 3. Example create message using the QName dialect

```
<s:Body>
  <wsrt:Create Dialect="http://schemas.xmlsoap.org/ws/2006/08/
    resourceTransfer/Dialect/QName"
    xmlns:d="http://example.org/sample">
    <ws mex:Metadata>
      <ws mex:MetadataSection Dialect="http://schemas.xmlsoap.org/
        ws/2006/08/resourceTransfer">
        <wsrt:Metadata>
          <wsrt:Lifetime>
          <wsrt:TerminateAt>
```

```

        <wsrt:TerminationTime>
            2006-04-11T12:00:00Z
        </wsrt:TerminationTime>
        <wsrt:CurrentTime>
            2006-04-10T10:00:54Z
        </wsrt:CurrentTime>
        </wsrt:TerminateAt>
    </wsrt:Lifetime>
    </wsrt:Metadata>
    </wsmex:MetadataSection>
</wsmex:Metadata>
<wsrt:Fragment>
    <wsrt:Expression>
        d:Volume
    </wsrt:Expression>
    <wsrt:Value>
        <d:Volume>
            <d:Drive>C:</d:Drive>
            <d:Label>MyDrive-C</d:Label>
            <d:TotalCapacity>10000000000</d:TotalCapacity>
        </d:Volume>
        <d:Volume>
            <d:Drive>D:</d:Drive>
            <d:Label>MyDrive-D</d:Label>
            <d:TotalCapacity>30000000000</d:TotalCapacity>
        </d:Volume>
    </wsrt:Value>
</wsrt:Fragment>
</wsrt:Create>
</s:Body>

```

The line:

```
<wsrt:TerminateAt>
```

indicates that after the resource is created, it is scheduled for destruction at the specified time. After this time, messages sent to the EPR and returned in the CreateResponse will fault. This line:

```
d:Volume
```

indicates that a resource is created with a specific value or set of values for the <d:Volume> property. This part of the code:

```

<wsrt:Value>
    <d:Volume>
        <d:Drive>C:</d:Drive>
        <d:Label>MyDrive-C</d:Label>
        <d:TotalCapacity>10000000000</d:TotalCapacity>
    </d:Volume>
    <d:Volume>
        <d:Drive>D:</d:Drive>
        <d:Label>MyDrive-D</d:Label>
        <d:TotalCapacity>30000000000</d:TotalCapacity>
    </d:Volume>
</wsrt:Value>

```

specifies the set of values of the `<d:Volume>` property. The response to this create message is illustrated in Listing 4:

Listing 4. Example CreateResponse

```
<s:Body>
  <wxf:ResourceCreated>
    <wsa:Address>http://www.example.org/diskport</wsa:Address>
    <wsa:ReferenceParameters>
      <xyz:ManagedResource>44355</xyz:ManagedResource>
    </wsa:ReferenceParameters>
  </wxf:ResourceCreated>
</s:Body>
```

The second through the seventh line in Listing 4 shows the EPR to the disk resource that is returned in the response message.

But wait! There's more!

Future *Meet the specs* columns on WS-RT 1.0 will look closer at how the WS-ResourceTransfer 1.0 specification extends the `put` operation. They will also detail:

- The `delete` operation
- The fault-handling rules of WS-Addressing
- Terminology and notation
- Security (how this WS works with WS-Security)

Resources

Learn

- Get information on the technologies discussed in this article: A working definition of [QName](#) and the [XML Path Language \(XPath\) 1.0](#).
- [WSDM/WS-Man Reconciliation: An Overview and Migration Guide](#): This whitepaper covers much of the current thinking on the immediate direction of the Web services management strategies.
- [Autonomic computing standards page](#): Get more information on the WS family of specifications.
- [Autonomic computing zone](#): Visit the developerWorks Autonomic computing zone to stay on the cutting edge of autonomic computing features and technologies.
- You might visit other developerWorks zones mentioned in this article: The [SOA and Web services zone](#) for the resources you need to understand and get started with SOA; the [XML zone](#) on how to manage, work with, and think in XML; the [Open source zone](#) to find resources for open source development and implementation.
- [developerWorks technical events and webcasts](#): Stay current with the latest technology.

Get products and technologies

- Grab your own [PDF](#) or [ZIP](#) version of the WS-RT 1.0 specification draft.
- [IBM trial software](#): Build your next development project with trial software, available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content](#).
- [developerWorks blogs](#): Get involved in the developerWorks community.

About the author

Kane Scarlett

Kane Scarlett is the editor of the Autonomic computing technology zone for developerWorks. His past publishing work was with such magazines as *Unix Review*, *Advanced Systems*, and the *-World* publications (*Java-*, *Sun-*, *NC-*, *Linux-*), as well as some little oddball journals like *National Geographic Magazine*.