[MS-FSRS]: Resource Store Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
11/06/2009	0.1	Major	Initial Availability
02/19/2010	1.0	Editorial	Revised and edited the technical content
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the Resource Store Protocol. This protocol enables a protocol client to store resources (typically files) on a protocol server. A typical scenario for using this protocol is an application creating resources that will be consumed by other applications.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Coordinated Universal Time (UTC)
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
resource
XML

The following terms are defined in [MS-OFCGLOS]:

HTTP GET
HTTP POST
URI (Uniform Resource Identifier)
URL (Uniform Resource Locator)
Web site

The following terms are specific to this document:

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary", June 2008.

1.3 Protocol Overview (Synopsis)

This protocol defines operations that allow a protocol client to upload, download, delete, verify for existence, query for the timestamp and list **resources**. A typical scenario for using this protocol is when one protocol client generates a resource that other protocol clients use at a later time.

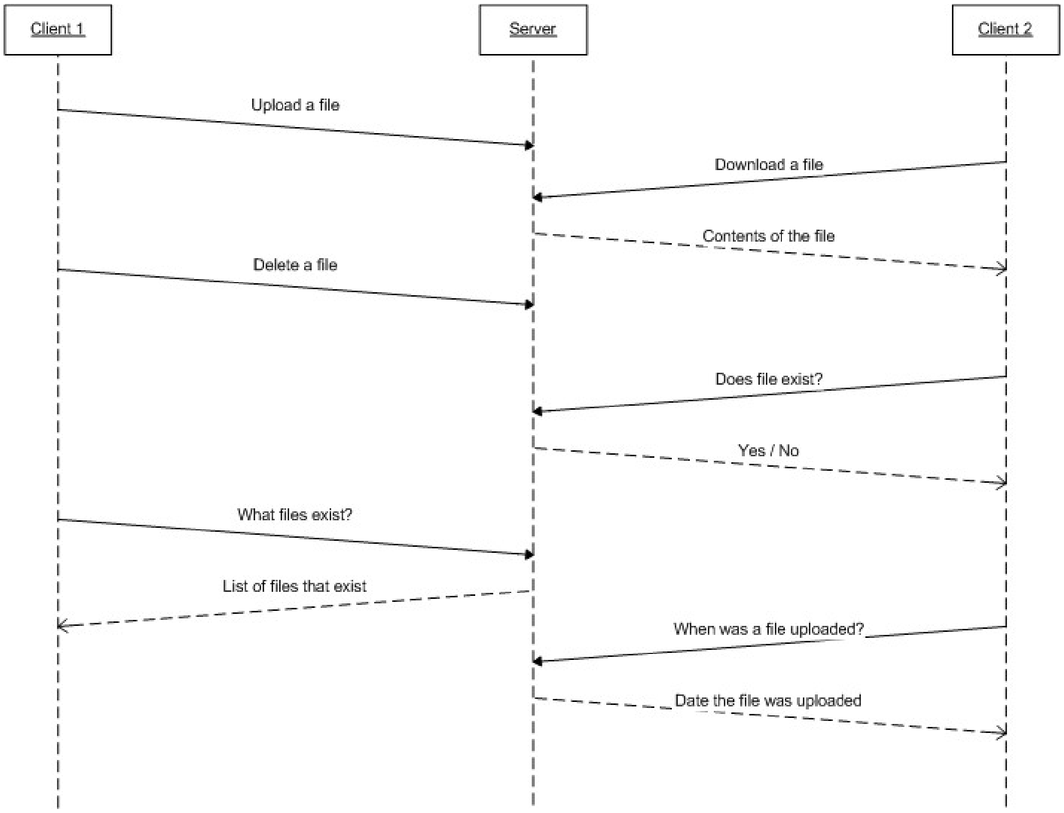

Figure 1: Communication between protocol clients and protocol server

1.4 Relationship to Other Protocols

This protocol uses **Hypertext Transfer Protocol (HTTP)** or **Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)** as shown in the following layering diagram:

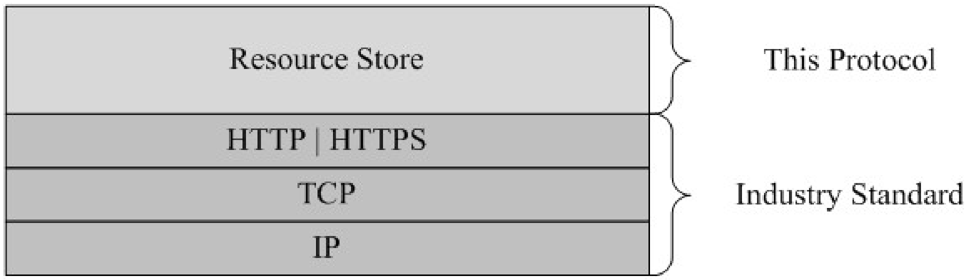

Figure 2: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

This protocol operates against a **Web site** that is specified by a **URL** that includes a port number that is known to protocol clients, for example http://www.contoso.com:13255.

This protocol assumes that authentication has been performed by the underlying protocols.

1.6 Applicability Statement

This protocol is designed for sharing data between protocol clients. Because the data is stored on the protocol server, this protocol is well-suited for protocol clients that can not connect to each other directly or simultaneously.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol MUST use HTTP for transport as specified in [RFC2616].

2.2 Message Syntax

2.2.1 Download

This retrieves a resource from the protocol server. It MUST be an **HTTP GET** request that contains a **URI** that represents an existing resource on the protocol server. The response uses HTTP return codes as specified in [RFC2616].

2.2.2 Exists

This verifies whether a specified resource is available on the protocol server. It MUST be an **HTTP POST** request as specified in [RFC2616], where the request-URI takes the following form:

```
//config/ListResources.aspx?OP=exists&URI=<resource’s URI string>
```

OP: Specifies the operation requested. It MUST contain a value of "**exists**".

URI: Specifies the URI of the resource on the protocol server.

These two parameters are required and can occur in any order but MUST be separated by an ampersand (**&**). The response uses HTTP return codes as specified in [RFC2616].

2.2.3 Upload

This adds a resource to the protocol server. It MUST be an HTTP POST request as specified in [RFC2616] where the request-URI takes the following form:

```
//config/ListResources.aspx?OP=upload&URI=<resources’s URI string>
```

OP: Specifies the operation requested. It MUST be **upload** for the **Upload** request.

URI: Specifies the URI of the resource on the protocol server. It MUST be an existing path (up to the resource name itself) on the protocol server.

These two parameters are mandatory and can occur in any order but MUST be separated by an ampersand (**&**).

The data sent in the request body MUST be the resource content.

The response uses HTTP return codes as specified in [RFC2616].

2.2.4 Delete

This request removes a resource from the protocol server. It MUST be an HTTP POST request as specified in [RFC2616] where the request-URI takes the following form:

```
//config/ListResources.aspx?OP=delete&URI=<resource’s URI string>
```

OP: Specifies the operation requested. It MUST be **delete** for the **Delete** request.

URI: Specifies the URI of the resource on the protocol server. It MUST represent an existing path on the protocol server.

These two parameters are mandatory and can occur in any order but MUST be separated by an ampersand (**&**).

The response MUST conform to HTTP return codes as specified in [RFC2616].

2.2.5 GetTimeStamp

This request retrieves the time that a specified resource on the protocol server was last modified. It MUST be an HTTP POST request as specified in [RFC2616] where the request-URI takes the following form:

```
//config/ListResources.aspx?OP=modified&URI=<resources’s URI string>
```

OP: parameter specifies the operation requested. It MUST be **modified** for the **GetTimeStamp** request.

URI: parameter specifies the URI of the resource on the protocol server. It MUST represent an existing path on the protocol server.

These two parameters are mandatory and can occur in any order but MUST be separated by an ampersand (**&**).

The response MUST conform to HTTP return codes as specified in [RFC2616]. See the following section for a successful response.

2.2.6 GetTimeStampResult

This message is the HTTP response to a **GetTimeStamp** request. It MUST contain an extra header field **X-Resource-Last-Modified** with the **Coordinated Universal Time (UTC)** of the last modification of the resource on the protocol server.

2.2.7 ListResources

The request retrieves a list of resources from a specified location on the protocol server. It also retrieves the associated date and type information. It MUST be an HTTP POST request as specified in [RFC2616] where the request-URI takes the following form:

```
//config/ListResources.aspx?OP=list&URI=<resources’s URI string>
```

OP: Specifies the operation requested. It MUST contain the value **list** for the **ListResources** request.

URI: Specifies the URI of the resource on the protocol server. It MUST represent an existing path on the protocol server.

These two parameters are mandatory and can occur in any order but MUST be separated by an ampersand (**&**).

The response MUST conform to HTTP return codes as specified in [RFC2616]. See the following section for a successful response.

2.2.8 ListResourcesResultSet

This message is the response to a **ListResources** request. The list of resources at a specified location is formatted in an **XML** document with the following Document Type Definition (DTD):

```
<!DOCTYPE resources [
    <!ELEMENT resources (resource*)>
    <!ELEMENT resource EMPTY>
    <!ATTLIST resource
        name CDATA #REQUIRED
        type CDATA #REQUIRED
        modifiedtime CDATA #REQUIRED>
]>
```

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, no additional timers or other state is required on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

3.1 Common Details

This section specifies details that are common to both protocol server and protocol client behavior.

Protocol clients SHOULD interpret HTTP Status-Codes returned by the protocol server as specified in [RFC2616] section 10.

This protocol allows protocol servers to perform various tasks and notify protocol clients of faults using HTTP Status-Codes.

3.2 Server Details

3.2.1 Abstract Data Model

The protocol client always initiates a request. A request always pertains to a resource on the protocol server. The protocol server always sends a response to a request.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Receiving a Download Message

The protocol server MUST respond with HTTP code 200 (OK) and return the resource as the data if the URI is valid.

The protocol server MUST respond with HTTP code 404 (not found) if the specified URI is invalid.

3.2.5.2 Receiving an Exists Message

The protocol server MUST respond with HTTP code 200 (OK) if the specified URI exists.

The protocol server MUST respond with HTTP code 404 (not found) if the specified URI is invalid.

3.2.5.3 Receiving an Upload Message

The protocol server MUST respond with HTTP code 200 (OK) if the specified URI is valid and the resource has been stored successfully.

3.2.5.4 Receiving a Delete Message

The protocol server MUST respond with HTTP code 200 (OK) if the specified URI is valid and the resource has been deleted successfully.

The protocol server MUST respond with HTTP code 404 (not found) if the specified URI is invalid.

3.2.5.5 Receiving a GetTimeStamp Message

If the specified URI is valid, the protocol server MUST respond with HTTP code 200 (OK) and include an extra header field **X-Resource-Last-Modified** with the Coordinated Universal Time (UTC) of the last modification of the resource on the protocol server.

The protocol server MUST respond with HTTP code 404 (not found) if the specified URI is invalid.

3.2.5.6 Receiving a ListResources Message

If the specified URI is valid, the protocol server MUST respond with HTTP code 200 (OK) and return a **ListResourcesResultSet** message as the data.

The protocol server MUST respond with HTTP code 404 (not found) if the specified URI is invalid.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

4.1 Download

A user wants to download a resource named ca_spell_is8859_.aut that is stored at /dictionaries/spellcheck/ on the protocol server.

Call:

```
GET //dictionaries/spellcheck/ca_spell_is8859_.aut
```

Result:

HTTP response code 200 (OK) and content.

4.2 Exists

A user wants to know whether a resource named ca_spell_is8859_.aut is stored at /dictionaries/spellcheck/ on the protocol server.

Call:

Note: This code example includes a line break to facilitate online and printed viewing. Remove the line break before running the code.

```
POST //config/ListResources.aspx?URI=/dictionaries/spellcheck/
ca_spell_is8859_.aut&OP=exists
```

Result:

HTTP response code 200 (OK).

4.3 Upload

A user wants to store a resource at the URI /dictionaries/spellcheck/ca_spell_is8859_.aut on the protocol server.

Call:

Note: This code example includes a line break to facilitate online and printed viewing. Remove the line break before running the code.

```
POST //config/ListResources.aspx?URI=/dictionaries/spellcheck/
ca_spell_is8859_.aut&OP=upload
```

Result:

HTTP response code 200 (OK).

4.4 Delete

A user wants to remove a resource stored at /dictionaries/spellcheck/ca_spell_is8859_.aut on the protocol server.

Call:

Note: This code example includes a line break to facilitate online and printed viewing. Remove the line break before running the code.

```
POST //config/ListResources.aspx?URI=/dictionaries/spellcheck/
ca_spell_is8859_.aut&OP=delete
```

Result:

HTTP response code 200 (OK).

4.5 GetTimeStamp

A user wants to know when the resource stored on the protocol server with URI /dictionaries/spellcheck/ca_spell_is8859_.aut was last modified.

Call:

Note: This code example includes a line break to facilitate online and printed viewing. Remove the line break before running the code.

```
POST //config/ListResources.aspx?URI=/dictionaries/spellcheck/
ca_spell_is8859_.aut&OP=modified
```

Result:

HTTP response code 200 (OK) with the following additional header field:

```
X-Resource-Last-Modified: 128832314677802079
```

4.6 ListResources

A user wants to know what resources are stored at /dictionaries on the protocol server.

Call:

```
POST //config/ListResources.aspx?URI=/dictionaries&OP=list
```

Result:

```
<xml>
<resources>
    <resource name="matching"
        modifiedtime="1/15/2009 10:27:40 AM" type="folder" />
    <resource name="spellcheck"
        modifiedtime="1/20/2009 10:44:31 PM" type="folder" />
    <resource name="spelltuner"
        modifiedtime="1/15/2009 10:27:40 AM" type="folder" />
</xml>
```

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® FAST™ Search Server 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index