# FTP RPC

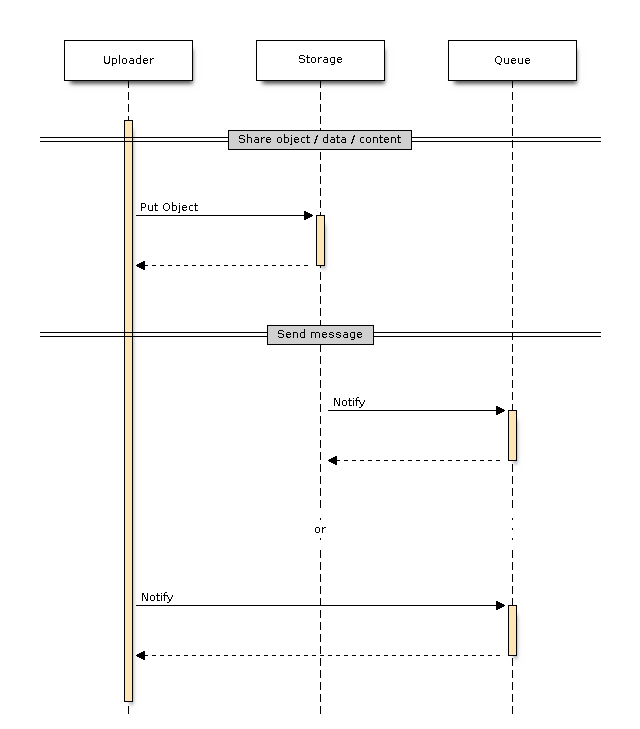
FTP via remote procedure call

Due to technical details of how the FTP and FTPS protocol are defined, systems that transfer files via FTP require a firewall with mostly open egress rules.

FTP RPC segregates the open egress requirement from the rest of the system(s).

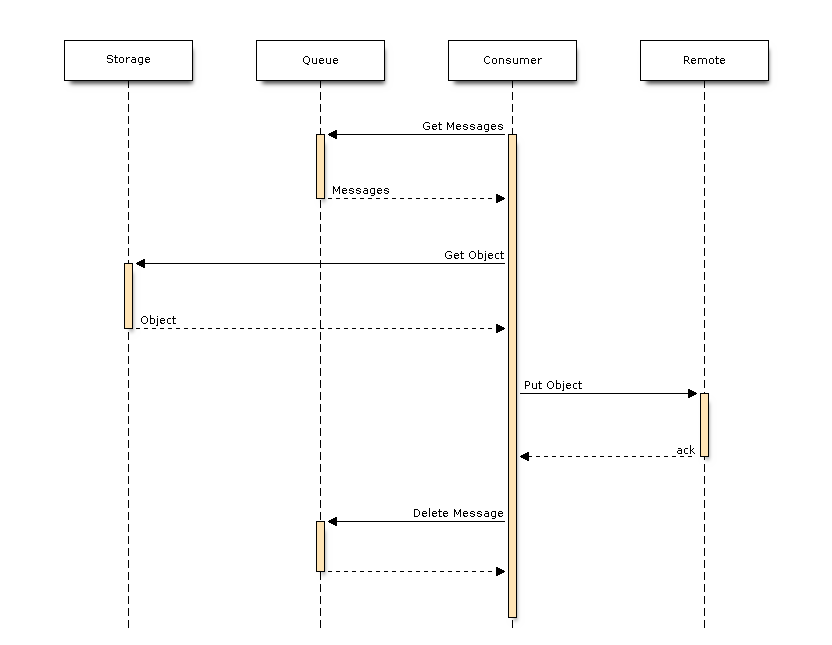
The system is asynchronous and does not return a response to the initiator.

## Outbound



| Name | Description |
| --- | --- |
| Uploader | Outbound initiator |
| Storage | Shared storage |
| Queue | Message queue |

### From Open Egress



| Name | Description |
| --- | --- |
| Storage | Shared storage |
| Queue | Message queue |
| Consumer | Message consumer |
| Remote | Remote / Vendor FTP |

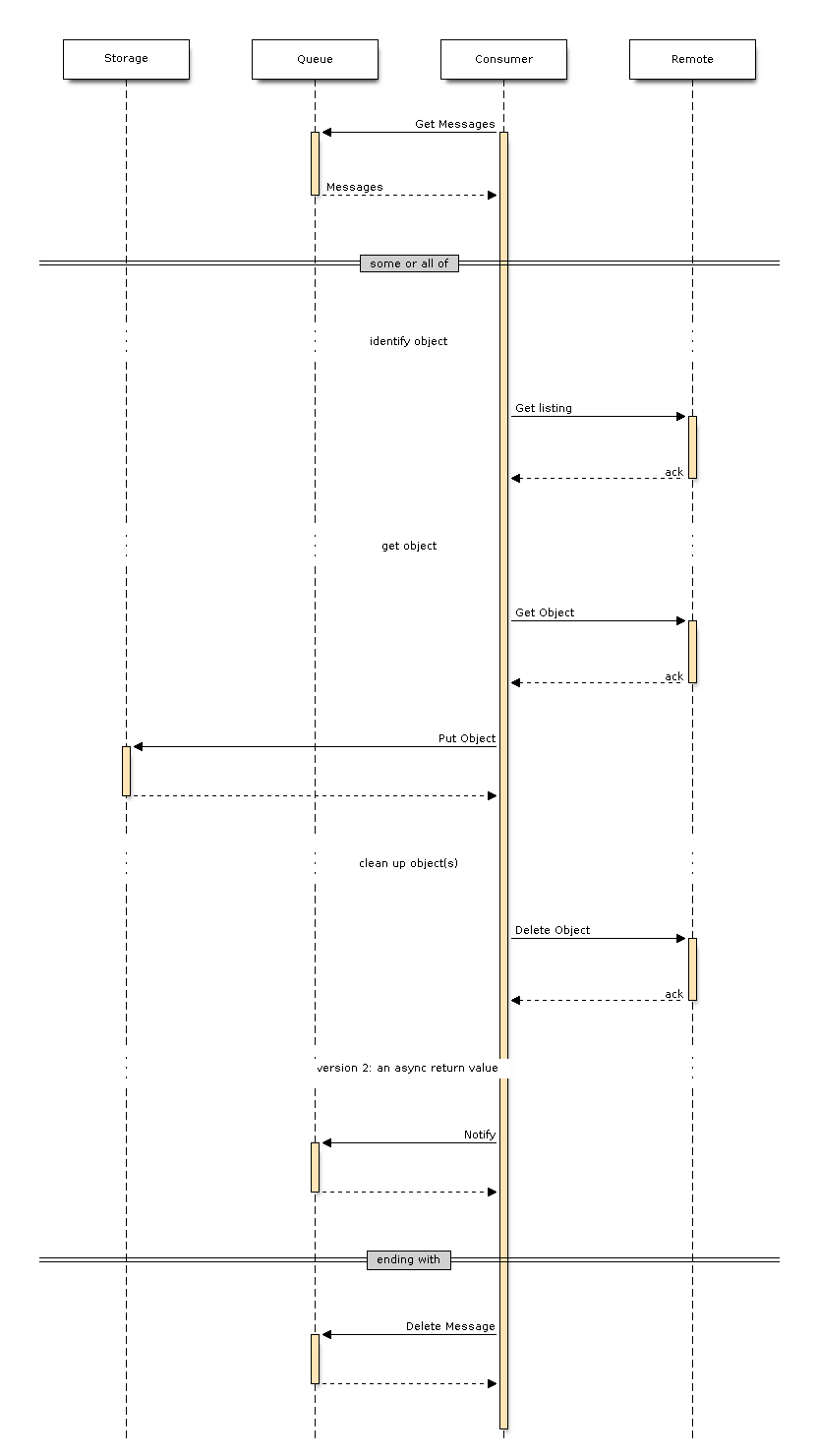
As of writing:

* “Shared storage” is AWS S3, which supports creating queue messages.
* “Message queue” is AWS SQS
* “Consumer” will be AWS Lambda

## Inbound

Starts with a request message sent describing the inbound request.

### From Open Egress



| Name | Description |
| --- | --- |
| Storage | Shared storage |
| Queue | Message queue |
| Consumer | Message consumer |
| Remote | Remote / Vendor FTP |

As of writing:

* “Shared storage” is AWS S3
  + AWS S3 supports creating queue messages on object creation events, delaying the need for consumer creating ack-notifications.
  + The use cases encapsulate the file-listing usages, meaning there is no need to send a “this is the file list” response.
* “Message queue” is AWS SQS
* “Consumer” will be AWS Lambda

For inbound requests, there is an expectation that there will be a later file-consumer that handles continuing the activates that happen after file has been acquired, either by queue-watching or storage-polling.

© Copyright OfficeSupply.com

Built with [Sphinx](http://sphinx-doc.org/) using a [theme](https://github.com/rtfd/sphinx_rtd_theme) provided by [Read the Docs](https://readthedocs.org/).