Using Fedora's Native Checksum Functionality

Background

The early versions of Fedora (up to 3.0) did not have any native checksum creation or testing capabilities[1]. We felt the need to use and test checksums early on, and so had to design and build our own system. The system we designed stores a SHA-1 checksum and a datastream ID for any archival datastream in the Fedora repository. The SHA-1 checksums are computed before ingest and stored in the technical metadata section of a foxml object. Once an object has been ingested, we routinely test its checksum using an external tool that parses the technical metadata section, streams the named archival datastream through the Fedora API[2], computes a SHA-1 checksum of the datastream, and compares this with the checksum stored in the object. While this system continues to work, it would be desirable to migrate to a system using Fedora's now robust native functions for the creation and testing of checksums.

Fedora's Current System

Fedora now provides the capability of computing and storing checksums for datastreams in a digital object, and later using these checksums to verify that the datastreams have not been altered in any way. This capability could be configured in fedora.fcfg to compute checksums automatically for all datastreams on ingest, or it could be used only for certain datastreams identified through a subelement of the datastreamVersion element, e.g.:

```
  <foxml:datastream ID="ARCH1" STATE="A" CONTROL_GROUP="M" VERSIONABLE="true">
    <foxml:datastreamVersion ID="ARCH1.0" LABEL="Jeffery&apos;s Book Test2 1/13/10: ARCH1"
CREATED="2010-01-13T10:10:10.0
00Z" MIMETYPE="application/x-tar">
      <foxml:contentDigest TYPE="SHA-256"
DIGEST="3c33e8b757a688949af171dd5f5aa271b83ec961fd4d209a743fcfc2cdac3685"/>
      <foxml:contentLocation TYPE="INTERNAL_ID" REF="rutgers-lib:25257+ARCH1+ARCH1.0"/>
    </foxml:datastreamVersion>
  </foxml:datastream>
```

The checksum, if pre-computed, can be stored in the DIGEST attribute as in the above example. If the DIGEST attribute is defined as "none" or simply left out, as in

```
<foxml:contentDigest TYPE="SHA-256"/>
```

the checksum will be computed by Fedora on ingest. If no DIGEST is specified, Fedora will read the file, compute the checksum, and store it in a new DIGEST

[1] Even when Fedora initially added automatically computer checksums, it did not yet have a working compareDatastreamChecksum method for testing these.

[2] This allows transparent handling of any kind of datastream, whether it is a managed content datastream ("M") or a redirected datastream ("R").

attribute of the internally stored FOXML. If a pre-computed checksum is specified, Fedora will still read the file, compute the checksum, and compare.

Checksums can also be defined in calls to the API-M functions addDatastream, modifyDatastreamByValue, and modifyDatastreamByReference using the checksumType and checksum parameters. Fedora supports the following checksum types: MD5, SHA-1, SHA-256, SHA-384, and SHA-512.[3] A value of "DISABLED" effectively turns off checksumming for a given datastream.

Checksums can also be computed and added to the foxml:datastreamVersion element using the REST API using a command like the following:

```
/usr/bin/curl -u "$fedoraAdmin:$fedoraAdminPasswd" -X PUT "http://$serverbase:8080/fedora/objects/rutgers-lib:123456/datastreams/ARCH1?checksumType=SHA-256"
```

Checksums stored this way in the foxml:datastreamVersion element can be tested using the API-M compareDatastreamChecksum method, which returns the checksum on success or the string "Checksum validation error" on failure.

Implementation Decisions

After considering our options, we have made a number of general implementation decisions.

Going forward we will use SHA-256 checksums, as these are more reliable than SHA-1 checksums and will not add too much overhead to the normal processing functions[4], though they will complicate the one time updating of legacy objects.

The WMS will continue to use pre-computed checksums in the foxml:contentDigest elements as a way of insuring that the datastreams are intact after ingest. As noted above, this will not adversely affect ingest times, as Fedora reads the file and computes a checksum whether or not a checksum is included in the DIGEST attribute of the foxml:contentDigest element.

As a continuing investigation, we will explore the possibility of using a time-based checksum testing cron rather than number of objects (e.g. 5 hours rather than x number of objects) and the possibility of running the nightly compare on staging where there are lots of extra cpu cycles.

---

[3] Note the attributes in the foxml:contentDigest element are different from the SOAP parameters.

[4] Recent timing tests suggest that sha-256 calculations take roughly twice the length of time needed for SHA-1. Fedora's Java method for comparing checksums is also considerably slower than the C program we currently use, but it has the benefit of offloading the functionality to native Fedora processes.

Migration Issues

Changing our system to make use of native Fedora checksum functions will have an impact on various parts of the RUcore project, including the WMS, dlr/EDIT, and signature testing.

WMS Impact

The WMS will have to begin including foxml:contentDigest elements within the foxml:datastreamVersion element for archival datastreams. The new checksums will use the SHA-256 rather than the SHA-1 algorithm. The WMS will no longer store checksums in the technical metadata sections, as this would create unnecessary redundancy. Legacy objects will continue to have checksums stored in technical metadata for the time being, which will allow us to continue our current form of testing while the data changes needed for native Fedora checksum testing are put in place.

As noted above, we have decided to have Fedora test our own pre-computed checksums to provide a valuable initial validation of the datastreams being ingested. Thus the WMS would continue to create pre-ingest checksums as before, only using the SHA-256 algorithm in place of SHA-1.

Dlr/EDIT Impact

The dlr/EDIT add and modify datastream functions will need to be updated to add the checksumType and checksum parameters for archival datastreams.

Signature testing Impact

The signature checking function will have to be changed to work with the Fedora API-M compareDatastreamChecksum method.[5] The approach would be to change the signature script so that it tests first to see if a new-style checksum is in place. If so, it would run the Fedora method; otherwise, it would run the old method using the SHA-1 signature in the TECHNICAL1 section. As more and more objects get the new signatures, the balance would shift toward greater use of the Fedora method. It should be noted that until the conversion is complete, replacing an archival datastream would require the manual deletion of the SHA-1 checksum from the TECHNICAL1 section.

We will need to come up with a method of updating earlier objects that do not have checksum information stored in foxml:contentDigest elements. Early tests suggested that this would need to be done with a command line script. There is currently no

[5] For the time being at least this method can still be used with the current alerting system. As it is a discrete function, compareDatastreamChecksum, will not interact automatically with Fedora's Java Messaging Service, and we will need to investigate any such connection separately.

API-M SOAP method that allows adding foxml:contentDigest elements without modifying the datastream itself. Recent tests, however, suggest that the REST API method outlined above could be run efficiently on a list of Fedora PIDs. This has the benefit of allowing Fedora itself to manage the datastream modifications and capture the modification events in its audit trail. While this is happening, a combination of traditional signature checking and the new Fedora checksum testing method could continue as described above. When the script has been run to add foxml:contentDigest elements with SHA-256 checksums to all older objects, the switchover to the new signature checking functions would be effected without interruption of the service. Note: the script should not be run until the WMS changes are in effect and SHA-256 checksums are being added routinely to newly ingested objects. This feature is expected to be in place for the R7.0 release. In the meantime, it can be thoroughly tested on objects in the rep-devel repository. After all legacy archival datastreams have been updated, the new signature checking functions can be put into operation, and the old TECHNICAL sections of these objects can be versioned, removing references to the old SHA-1 checksums, with a fixds project. Note that a separate specification will be developed detailing the process for updating the legacy objects to SHA-256, removing the techMD checksums, and possibly unpacking the legacy tar files into separate archival master datastreams for each file encapsulated in the tar.

JAT (September 11, 2012)