Resolving References in Distributed Objects

Release Date: 07/24/2003

Author: Chris Vicary and Andrea Goethals, FCLA

Change History:

The FCLA defines a distributed object as “a set of data files such that one of them, the root data file, contains embedded links to the other data files. If the file name of a data file changes, the root data file must also be changed. A distributed object may or may not correspond to a logical object. A distributed object may contain other distributed objects, for example a web page that links to a second web page that also contains links.” XML documents and XML DTD documents are examples of file formats that may be root data files of distributed objects. This report describes the algorithm that will be used by the FDA to locate the physical files referred to by root data files.

Files that are capable of being root data files of distributed objects will be parsed to determine if they do contain named references ('links') to other files. Each link will be characterized by four variables:
1.The type of URI
2.The origin of the data file containing the link
3.Whether or not the archive was given a checksum for the linked-to file.
4.Whether or not the link is to a file deemed essential by the archive to the referencing data file's integrity (ex: schema).

The combination of these variables will determine how the link is resolved by the archive program.

Type of URI
The are four types of URIs that DAITSS will recognize:
HTTP_URL - A URL using the http protocol. Ex: “http://www.example.org/a.pdf”
REL_PATH - A relative path. Ex: “images/1.jpg”
ABS_PATH - An absolute path within a file system. Ex: “/myfiles/1.jpg” or “C:\myfiles\1.jpg”
OTHER - All other URIs that do not fit one of the above three patterns. Ex: ftp://www.example.org/a.pdf”

The origin of the data file containing the link
DAITSS will characterize a data file's origin as one of three values:
CUSTOMER - A file already part of the data package before archive ingest begins.
ARCHIVE - A file created by DAITSS. Ex: SIPDescriptor
INTERNET - A file downloaded from the Internet by DAITSS.

Checksum Availability
DAITSS will record one of two values for this category for each link:
CHECKSUM - A checksum value is available for the linked-to file.
NO_CHECKSUM- A checksum value is not available for the linked-to file.

Link Importance
DAITSS will record one of two values for this category for each link:
NEEDED - The archive thinks that the link is essential to interpreting the data file's contents.
NOT_NEEDED - The archive thinks that the link is not essential to interpreting the data file's contents.

Link Resolution:

URI Type	*Data file Origin*	*Have Checksum*	*Link Importance*	*Algorithm*
HTTP_URL	INTERNET	NO_CHECKSUM	NEEDED	Download the file from the Internet. If the file can't be downloaded - record link as a 'Broken Link' in the database.
HTTP_URL	INTERNET	NO_CHECKSUM	NOT_NEEDED	Record link as an 'Ignored Link' in the database.
HTTP_URL	INTERNET	CHECKSUM	NEEDED or NOT_NEEDED	(IMPOSSIBLE)
HTTP_URL	ARCHIVE	NO_CHECKSUM	NEEDED or NOT_NEEDED	(WON'T LET HAPPEN)
HTTP_URL	ARCHIVE	CHECKSUM	NEEDED or NOT_NEEDED	(IMPOSSIBLE)
HTTP_URL	CUSTOMER	NO_CHECKSUM	NEEDED	1. Extract file name from URL. 2. Search the referencing data file's directory for the file name. If it is there, the file is found. If not, continue. 3. Search package directory and its subdirectories for the file name. If one matches, it is found. If more than one matches, log as an warning and stop (DAITSS MultipleFilesFoundWarning). If there are no matches, continue. 4. Download the file from the Internet. If the file can't be downloaded - record link as a 'Broken Link' in the database .
HTTP_URL	CUSTOMER	NO_CHECKSUM	NOT_NEEDED	(WON'T LET HAPPEN)
HTTP_URL	CUSTOMER	CHECKSUM	NEEDED	1. Extract file name from URL. 2. Search package directory and its subdirectories for the file name 3. For any file name matches calculate the MD5 checksum of the files. 4. If any checksums match the given checksum, the file is found. If no checksums match or there were no file name matches - record link as a 'Broken Link' in the database .
HTTP_URL	CUSTOMER	CHECKSUM	NOT_NEEDED	(WON'T LET HAPPEN)
REL_PATH	INTERNET	NO_CHECKSUM	NEEDED	1. Convert relative path to URL 2. Download the file from the Internet. If the file can't be downloaded - record link as a 'Broken Link' in the database .
REL_PATH	INTERNET	NO_CHECKSUM	NOT_NEEDED	Record link as an 'Ignored Link' in the database.
REL_PATH	INTERNET	CHECKSUM	NEEDED or NOT_NEEDED	(IMPOSSIBLE)

URI Type	*Data file Origin*	*Have Checksum*	*Link Importance*	*Algorithm*
REL_PATH	ARCHIVE	NO_CHECKSUM	NEEDED	Look for the file using the relative path in relation to the directory location of the referencing file. If it is found there, it is the file. If it is not found there, throw a DAITSS PathNotFoundException. (Archive program halts).
REL_PATH	ARCHIVE	NO_CHECKSUM	NOT_NEEDED	(WON'T LET HAPPEN)
REL_PATH	ARCHIVE	CHECKSUM	NEEDED or NOT_NEEDED	(IMPOSSIBLE)
REL_PATH	CUSTOMER	NO_CHECKSUM	NEEDED	Look for the file using the relative path in relation to the directory location of the referencing file. If it is found there, it is the file. If it is not found there - record link as a 'Broken Link' in the database . (note - there was discussion about whether we should also look in the referencing file's directory.)
REL_PATH	CUSTOMER	NO_CHECKSUM	NOT_NEEDED	(WON'T LET HAPPEN)
REL_PATH	CUSTOMER	CHECKSUM	NEEDED	1. Extract file name from the relative path. 2. Search package directory and its subdirectories for the file name 3. For any file name matches calculate the MD5 checksum of the files. 4. If any checksums match the given checksum, the file is found. If no checksums match or there were no file name matches - record link as a 'Broken Link' in the database .
REL_PATH	CUSTOMER	CHECKSUM	NOT_NEEDED	(WON'T LET HAPPEN)
ABS_PATH	INTERNET	NO_CHECKSUM	NEEDED	Record link as a 'Broken Link' in the database.
ABS_PATH	INTERNET	NO_CHECKSUM	NOT_NEEDED	Record link as an 'Ignored Link' in the database.
ABS_PATH	INTERNET	CHECKSUM	NEEDED or NOT_NEEDED	(IMPOSSIBLE)
ABS_PATH	ARCHIVE	NO_CHECKSUM	NEEDED or NOT_NEEDED	(WON'T LET HAPPEN)
ABS_PATH	ARCHIVE	CHECKSUM	NEEDED or NOT_NEEDED	(IMPOSSIBLE)
ABS_PATH	CUSTOMER	NO_CHECKSUM	NEEDED	1. Extract file name from absolute path. 2. Search the referencing data file's directory for the file name. If it is there, the file is found. If not, continue. 3. Search package directory and its subdirectories for the file name. If one matches, it is found. If more than one matches, log as an warning and stop (DAITSS MultipleFilesFoundWarning). If there are no matches, record link as a 'Broken Link' in the database .
ABS_PATH	CUSTOMER	NO_CHECKSUM	NOT_NEEDED	(WON'T LET HAPPEN)

URI Type	***Data file Origin***	***Have Checksum***	***Link Importance***	***Algorithm***
ABS_PATH	CUSTOMER	CHECKSUM	NEEDED	1. Extract file name from the absolute path. 2. Search package directory and its subdirectories for the file name 3. For any file name matches calculate the MD5 checksum of the file(s). 4. If any checksums match the given checksum, the file is found. If no checksums match or there were no file name matches - record link as a 'Broken Link' in the database .
ABS_PATH	CUSTOMER	CHECKSUM	NOT_NEEDED	(WON'T LET HAPPEN)
OTHER	INTERNET or ARCHIVE or CUSTOMER	CHECKSUM or NO_CHECKSUM	NEEDED or NOT_NEEDED	Record link as an 'Ignored Link' in the database.