The International Journal of Digital Curation
Issue 2, Volume 6 | 2011

Automating the Extraction of Metadata from Archaeological Data Using iRods Rules

David Walling & Maria Esteva,
Texas Advanced Computing Center

Abstract

The Texas Advanced Computing Center and the Institute for Classical Archaeology at the University of Texas at Austin developed a method that uses iRods rules and a Jython script to automate the extraction of metadata from digital archaeological data. The first step was to create a record-keeping system to classify the data. The record-keeping system employs file and directory hierarchy naming conventions designed specifically to maintain the relationship between the data objects and map the archaeological documentation process. The metadata implicit in the record-keeping system is automatically extracted upon ingest, combined with additional sources of metadata, and stored alongside the data in the iRods preservation environment. This method enables a more organized workflow for the researchers, helps them archive their data close to the moment of data creation, and avoids error prone manual metadata input. We describe the types of metadata extracted and provide technical details of the extraction process and storage of the data and metadata.[1]

[1] This paper is based on the paper given by the authors at the 6th International Digital Curation Conference, December 2010; received December 2010, published July 2011.
The *International Journal of Digital Curation* is an international journal committed to scholarly excellence and dedicated to the advancement of digital curation across a wide range of sectors. ISSN: 1746-8256 The IJDC is published by UKOLN at the University of Bath and is a publication of the Digital Curation Centre.

Introduction

The Institute of Classic Archaeology (ICA), a research unit at the University of Texas at Austin, has been conducting various overseas archaeological projects involving specialists in several countries. Beginning in 2000, digital data has been continuously generated but loosely organized, creating serious bottlenecks in the research process and limiting data reuse.

The Texas Advanced Computing Center (TACC) is a research and services unit at the University of Texas at Austin with a mission to enable discoveries that advance science and society through the application of advanced computing technologies. Since 2008, the Data Management and Collections group at TACC has worked with diverse research collections in areas such as herbarium, art, sociology, and economics. The group builds and maintains large data-management and storage resources, and consults with collections' creators in all aspects of the data lifecycle, from creation to long-term preservation and access.

An ongoing collaboration between TACC and ICA to manage and archive ICA's evolving data collection resulted in a method to automate the capture of metadata from the collection's record-keeping system during ingestion into a preservation and management environment. This metadata provides the necessary contextual and versioning information to render a history of the research process (Esteva et al., 2010). The system involves the implementation of a record-keeping system, utilization of Integrated Rule Oriented Data System[2] rules to execute a custom extractor script, and encoding of information in metadata schema standards.

In this paper, we begin by providing a background to the aggregation of archaeological data and describing the data management problems that happen throughout the process of documenting archaeological research. Next, we explain how the file naming and directory hierarchy conventions help to streamline the process of classifying the data objects so that the relationships between them are preserved. After describing how the information implicit in this system is mapped to appropriate metadata schemas, we provide an overview of the task workflow for extracting that metadata, in addition to extracting metadata from additional sources using iRods rules and a Jython script. We then provide technical details of how this workflow is implemented. We conclude by describing the client interfaces for accessing the extracted metadata and discuss future challenges.

Archaeological Data: Problems Managing Research Documentation

The act of excavation inherently destroys the archeological site. This makes it critically important to document the site in as much detail as possible and to preserve that documentation with utmost care. With an increased reliance on tools such as digital cameras and 3D modeling applications, knowledge of the site is increasingly being created in a digital form. This knowledge is generated at various stages of the research process, from fieldwork to analysis and final publication. The standard archeology documentation practice is to record the data in relationship to the stratigraphic unit or "context" from which the objects are excavated, and to reflect the

[2] iRods Version 2.3: http://www.irods.org.

stage in which data is produced. Data objects are related by their association to the same context, or because they are documentation about the same object at different stages of the research process (e.g., at the site, before or after conservation). While archeology documentation standards stress that these relationships are to be recorded consistently, differing practices and the distributed character of archeology research teams has resulted in the production of extensive and complex digital data which, in general, tends to be managed haphazardly, with little metadata and lacking systematic organization (Esteva et al., 2010).

In the case of ICA, in addition to the challenges described above, there was a need to provide a connection between digital objects existing in their native formats to representations of those objects entered by field workers into the Archaeological Recording Kit[3] database system. While ARK is used to manage the research data and its public presentation, it does not address the long-term preservation of the raw data which is extensively used for detailed analysis and for publication. It is in ARK where object descriptions and context relationships are recorded, but the raw data objects themselves are represented only by an identification number and versions of the objects in lower quality data formats. ICA researchers stressed the need for ensuring the long-term preservation of the raw data, their descriptions, and the relationships between associated data objects.

A Record-Keeping System to Classify and Describe Archaeology Data

The first step of the collaboration was to appraise and inventory ICA's data collection. The data was accumulated over the course of ten years and stored on two data servers and various personal computers and storage devices in the US and abroad. The data, which documents the objects found on the site at the different stages of the archeology research process, consists of born digital and digitized objects including: digital photography of context areas and artifacts; digitized paper drawings, field notes, context sheets and register forms; 3D models; GIS data; specialist's reports, analysis tables, and final publications.

Based on the assessment and the study of the workflows that generate data at the different stages of the research process, the team decided to implement a record-keeping system which consists of: a) an improvement over the existing file naming convention used in the field for the data and b), a hierarchically labeled directory structure to classify and logically store data by type of archaeology object and documentation type. In conjunction with technical metadata extracted from the data objects, the record-keeping system provides the needed contextual metadata about the objects and their relationships, as well as information needed for the data's long-term preservation.

While this record-keeping system still requires archaeologists to manually name files and place them in the directory hierarchy appropriately, it reduces the time cost in several ways. By placing a given data object in a particular folder, that object's metadata will reflect its association with all categories implicit in the directories path, thus preventing the need to add each category by hand at a later time. Moreover, simply moving a misplaced file to a new location in the hierarchy automatically re-

[3] ARK Version 0.6: http://ark.lparchaeology.com/.

categorizes the object accordingly. Naming an object in accordance with the convention allows for the automatic extraction of metadata about that object from other systems, primarily ARK, which prevents the need to manually gather that information for each object in the collection. Since this is a manual step, the extraction process must account for human error, which we address in future sections.

Directory Hierarchy

Figure 1 below shows a partial view of the directory structure, highlighting the hierarchy created to represent the types of image documentation that are gathered about special finds in the field. The hierarchical directory structure serves to categorize the data as they are gathered and produced in the different research stages, close to the point at which they were created. Top-level directories are labeled according to documentation type. For each type, the sub-directories within reflect the materials subject to that type of documentation, and then the different kinds of documentation that are generated during further study of these materials. Hierarchal categorization is a familiar and common way for users to organize data objects (Henderson, 2009).

File Naming Convention

The file naming convention provides four primary pieces of information: the key for the archaeological object represented by the data object as it is documented in ARK, nomenclatures used by archaeologists to designate objects as they are found, the stage of the research process in which the data object was produced, and a designation as either a master or altered version. This structured formatting of the file name allows programmatic splitting of the name into its logical parts during the metadata extraction phase.

Illustrative Example.

Taking the data object named ‘sfi_CH01PA_8_a1_m.JPG’ as an example, ‘sfi_CH01PA_8’ matches the object key in ARK, ‘a1’ designates that it was produced *after conservation*, and ‘m’ indicates that it is the original version of the file. Multiple data objects may be produced for a given archaeological object at a given research stage, so a sequence is recorded numerically following the stage designation. The data object ‘sfi_CH01PA_8_a2_m.JPG’ is another image created *after conservation* representing the archaeological object ‘sfi_CH01PA_8’. The research stage designations were developed by the researchers and include: ‘b’=before conservation,‘d’=during conservation, ‘a’=after conservation, ‘l’=lifting, ‘m’=microscope, and ‘s’=studio.

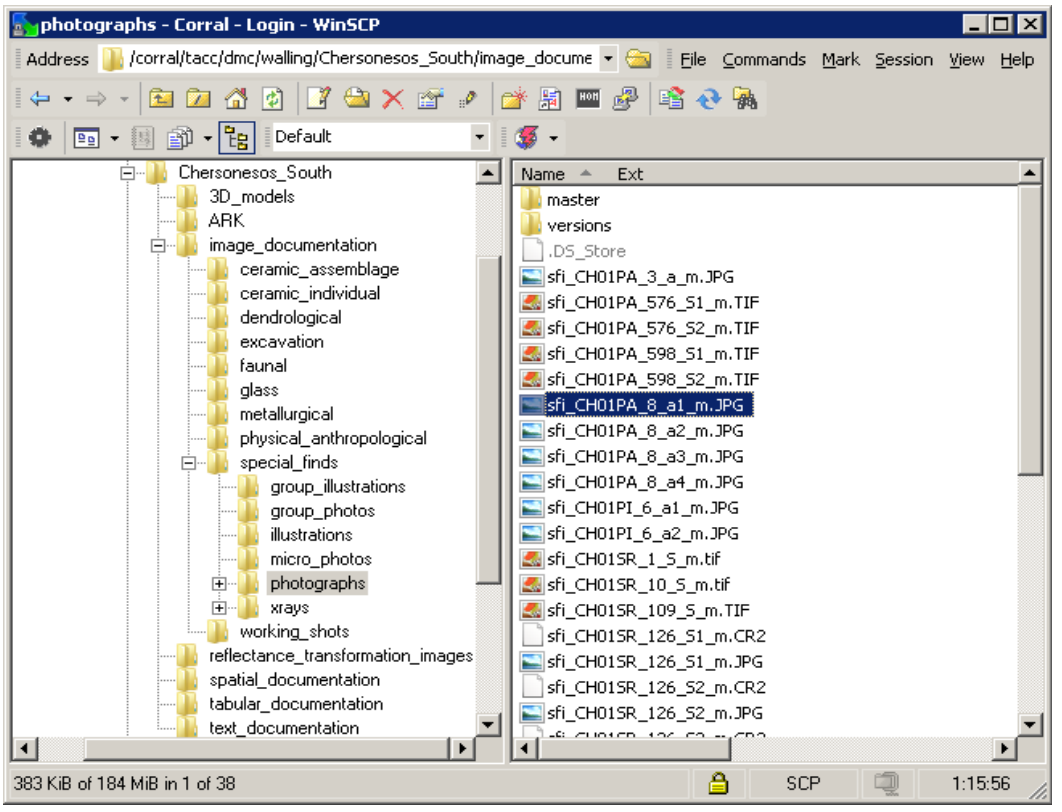

Figure 1. Example of the file naming convention and directory hierarchy for an image of a special find.

Preservation and Management Environment

The archived data collection is hosted within an instance of iRods running on TACC's storage application resource Corral. Corral consists of 1.2 PB of online disk space and a number of servers providing high-performance storage, database, and web hosting. This allows for all the applications related to the project, such as the ARK database, web-interfaces and a GIS framework to exist within the same architecture. A high-performance parallel file system is accessible from TACC's computational resources, enabling seamless paths between storage and analysis of potentially petabyte-scale datasets. Researchers at the University of Texas at Austin can request a storage allocation in Corral and work in that environment to develop their collections. TACC maintains the systems and provides consulting as needed by the user.

iRods is a data management platform for automating tasks important to long term preservation of data, such as off-site replication to heterogeneous storage systems, data integrity and authenticity verification, and format migration. These tasks are controlled by iRods' "rule engine" allowing for administrators and end users to specify management tasks to be executed for a given conditional trigger. For example, a rule might be to automatically replicate a data object to two offsite storage locations at the time of ingestion. It is this rule engine that enables us to plug in a custom script for metadata extraction. For this project, iRods will execute our metadata extractor script each time an object is added to, renamed, moved, or removed from the collection. Metadata extracted by this script is entered into the iRods Catalog (iCat) providing queryable information for items in the collection.

Mapping the Metadata

A data dictionary was created to map the file naming and record-keeping information to qualified Dublin Core (DC-Q). DC-Q metadata is focused on describing data objects, providing information useful for finding objects of interest as well as gaining understanding of what the objects represent and their relationship to other objects. Each folder in a data object's path represents a category or keyword relevant to the object and is mapped to the *subject* field of DC-Q. The file name provides the *identifier*, as explained previously, and the portion specifying the stage of research is mapped to the *relation->ispartof* DC-Q field. These mappings allow for recording relationships between data objects explicitly within the metadata, making it possible to write queries against the fields. For example, a user can search these fields to find all data objects of a given category (*subject*) associated with a given archaeological object (*identifier*) created at a given stage of the research process (*relation->ispartof*).

In addition to the metadata provided by the record-keeping system, the ARK object key explicitly recorded in the file name allows us to extract additional information from the ARK database, when that information exists. This may include information such as the archaeological context to which the objects belong, descriptive summaries for the objects, and excavation dates. The context identifier is mapped to *relation->ispartof*, and serves as one of the most fundamental relationships between objects in the collection.

Technical metadata, critical to the long-term preservation of digital objects, is generated using the File Information Tool Set[4] and mapped to the Preservation Metadata: Implementation Strategies schema (PREMIS). FITS aggregates various tools including Jhove, Exiftool, Droid, and NLNZ Metadata Extractor. FITS outputs to an XML record, which includes metadata such as the file type, file format version, file size, creating application, MD5 checksum, and file dependent metadata such as image resolution.

Descriptive and technical metadata are encoded in a Metadata Encoding and Transmission Standard document (METS). This document is stored alongside the original data object in iRods, with a subset of the metadata being registered in iCat. By extracting metadata into METS, we are able to decouple contextual and descriptive information from both ARK and our own record keeping system and store it in a form more suitable for preservation and sharing. Figure 2 below shows parts of a generated METS document for a sample data object.

```
- <mets:dmdSec ID="DM1">
  - <mets:mdWrap MDTYPE="DC" LABEL="Dublin Core Qualified Metadata">
    - <mets:xmlData>
        <dcterms:title>sfi_CH04SR_2143_i.jpg</dcterms:title>
        <dcterms:publisher>ICA</dcterms:publisher>
        <dcterms:spatial>Chersonesos</dcterms:spatial>
        <dcterms:temporal>Hellenistic</dcterms:temporal>
        <dcterms:isPartOf>CH04SR_341</dcterms:isPartOf>
        <dcterms:description>short_desc: glass fragment</dcterms:description>
```

Figure 2. Example METS document, highlighting parts of the DC-Q section.

[4] FITS Version 0.4.2: http://code.google.com/p/fits/.

Task Workflow

The primary iRods rule created for this work governs the actions taken when a new object is ingested to the collection. In this case, a Jython script is called that orchestrates numerous sub-tasks for extracting metadata. These steps are outlined in Figure 3. Similar rules handle additional data object manipulations, each utilizing the Jython script to accomplish the appropriate tasks. While archives are inherently static, data object removal, renaming, or relocation may be necessary, such as in cases of misclassification of a file within the directory hierarchy. These rules enable controlled manipulation of the data objects. For example, enforcing administrative approval for the removal of a file inadvertently added to the archive.

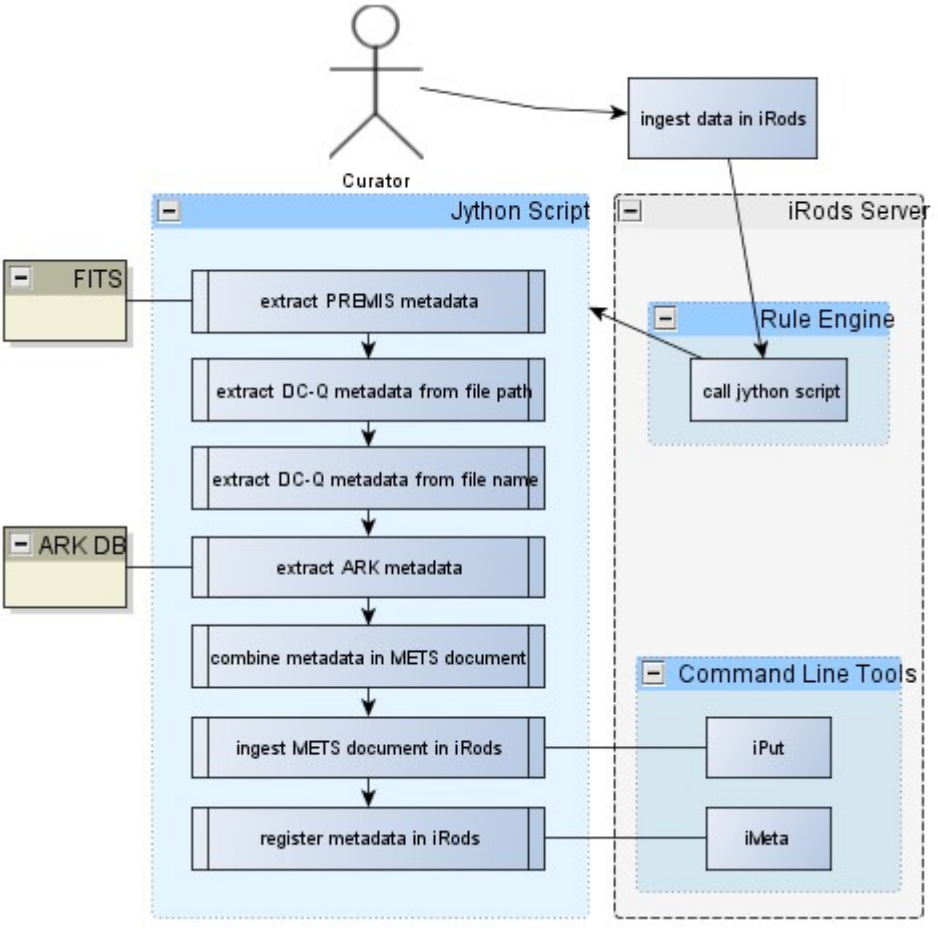

Figure 3. Task workflow diagram. Explanations for each step are provided below.

Workflow Implementation

The implementation of the above workflow involved the creation of the metadata extractor script and the integration of this script into the iRods rule engine. Here we describe this implementation, highlighting the challenges, and we make the actual code available for further review in the references.

Metadata Extractor Script

The metadata extractor script was implemented in Jython v2.6, a Java based implementation of Python. Jython was chosen as it provides both access to Python modules for performing string manipulation, regular expressions, reading and writing files, executing system commands, and querying databases, as well as Java's superior XML and XSLT tool libraries. It was the desire to leverage advanced XSLT 2.0 features that required us to choose a Jython implementation, as Python modules for those features do not yet exist. Jython scripts are executed within the Java Virtual Machine (VM), incurring VM startup time costs, which under certain usage patterns can be prohibitive. Care was taken to ensure that the execution of the script was designed to minimize this cost, mainly by designing the script to handle multiple data object processing in a single execution, as opposed to executing the script multiple times.

Two parameters are passed to the script when it is executed by the iRods rule engine: the path of the folder or file triggering the rule execution and the iCommand being executed. iCommands are iRods' versions of the standard Unix tools for file management, such as mv, ls, cp, put, and rm. iCommands are executed either directly by an end user, or on behalf of the user via an iRods GUI client, such as iRods Explorer. Our script performs various management tasks for irsync, iput, icp, irm, and imv.

Ingestion.

Both iput and icp essentially create a new data object in the iRods collection. Since the end result of our script is to create a METS encoded XML document containing the extracted metadata and to store that file in iRods, our first step is to ensure that the data object triggering the script is not one of these METS files, otherwise we would create an infinite loop. Naming these METS documents with a consistent appendix of '.mets.xml' allows us to easily filter them out using regular expressions.

We then proceed by executing the FITS tool and outputting the resultant XML to a temporary location on the file system. An XSL stylesheet maps the content of the FITS output to PREMIS and DC-Q fields and, after being transformed via XSLT, our initial METS encoded XML document exists with a subset of the complete metadata in place. In addition to using standard xpath expression in our XSL file to locate the appropriate content in the FITS output, we utilized a suite of custom XSLT functions provided by FunctX[5]. One of the trickier aspects of getting the XSLT transformation working properly was ensuring that we carefully defined the various schemas in use (mets, dcterms, dctype, fits, and functx) and provided the correct schema location. Additionally, our resultant METS document does not specify any custom namespace, so the document can validate directly against the standard schemas in use and we avoid the need to provide our own schema definition (xsd).

[5] FunctX: http://www.functx.com.

Next, we extract metadata from the file naming and directory hierarchy convention utilizing various Python modules and insert this information into our newly created METS document. The directory path is split into its various nodes and each are inserted into the document as *<dcterms:subject/>* elements. The '_' character used in the file name allows the name to be split into its various logical components, with the research stage and master-versus-altered codes being converted to human readable representations. We utilize the JDOM API libraries to perform our XML document manipulation, which was another driver for selecting Jython over Python for our implementation.

Using the object key extracted from the file name, we are able to query the ARK database using zxJDBC[6]. ARK is where the relationship between the physical object and the archaeological context is stored, and we explicitly add this information to the METS document DC-Q fields, when available. If the context relationship cannot be determined, we utilize a special token 'Context Unavailable' in its place. This allows users to search for items unassociated with a given context, either to check for information possibly missing in ARK, or to find objects legitimately outside the scope of a given context. An example of the latter might be a report dealing with multiple contexts, with the relationship to those various archaeological contexts having never been recorded and outside the scope of the system.

At this point, we have extracted as much metadata about the object as there is available, and we now want to make a subset of that metadata available in iCat where it can be used to search for objects of interest. We accomplish this using xpath to extract the descriptive data from the DC-Q section of the METS document and register it in iCat using the imeta iCommand. We simply format the information extracted from METS as key/value pairs, where key equals the DC-Q element name and value equals the XML nodes value. We then pass this pair in a system execution of imeta.

Finally, we execute the iput command to ingest our newly created METS document into iRods, storing it in the same location as the data object which it describes.

Moving, Renaming, and Removal.

Moving a file to a new location in the directory hierarchy or renaming the file essentially changes the metadata for that object. In these events, we remove the previously created METS document from iRods, clear the iCat entries for the object, and recreate the metadata based on the new information. By design, iRods does not hard delete a file when it is removed. Rather, the file is moved to a special trash folder where the file can be recovered at a later time. On removal, we provide logic to remove the associated METS document as well, with iRods handling the clearing of the iCat metadata for us.

Error Handling.

In cases where the file name does not conform to the file naming convention, a large amount of metadata is still generated for the data object based on the FITS output and the location within the directory hierarchy. This requires care in checking for and handling exceptional cases within the Jython script appropriately. Since archaeological context association is such a fundamental relationship for objects in the collection,

[6] zxJDBC Version 2.1: http://sourceforge.net/projects/zxjdbc/.

objects for which this cannot be determined explicitly add a *relation->ispartof* value of 'Context Unavailable'. This allows for users to find all such objects for review.

iRods Integration

The primary challenge when integrating the extractor script with iRods was gaining a thorough understanding of the iRods rule engine. In particular, figuring out how to make a Jython script call from a rule, passing along the appropriate parameters. Rules are defined in .irb files in the iRods server's configuration directory, and are registered in *server.config* in the *reRuleSet* property. By default, iRods will only execute one rule for a given matching trigger. This is not always desirable. For example, on file ingestion we want system-wide administrative rules, such as off-site replication, to occur in addition to executing our metadata extractor script for items in this particular collection. iRods enables this type of scenario utilizing the *applyAllRules()* rule function, although we found the usage of this function to be poorly documented. We refer the reader to our referenced code (Walling & Esteva, 2010) for examples of how to use this function.

As mentioned previously, iRods does not delete files on removal, instead it moves them to a special trash folder. This results in iRods calling the same rule type for both an *imv* and *irm*. In our case, we needed to perform different tasks in these two scenarios, and thus we needed to know which iCommand had triggered the rule execution. This is accomplished by adding a conditional statement in the rule definitions to check if the file is being sent to the trash, indicating that irm was called. We refer the reader to our code (Ibid) and the iRods documentation for more information on implementing rules.

Client Interfaces

With the metadata registered in iCat, end users can utilize a wide variety of iRods client interfaces to search the collection for data of interest. An example use case would be to find all original photographic documentation for a given archaeological context identifier. Previously, users could only find lower quality versions of this knowledge as it exists in ARK, with no way to link to the higher quality original documentation. This type of flexible query ability enables use cases not supported by ARK, namely to find all research documentation not existing in ARK.

These interfaces are also used to manage existing data and add new objects to this evolving data collection. Command line, GUI, and WebDav interfaces are available and users can utilize the one with which they feel most comfortable, with the underlying execution of the metadata extraction and archiving tasks being independent of their choice.

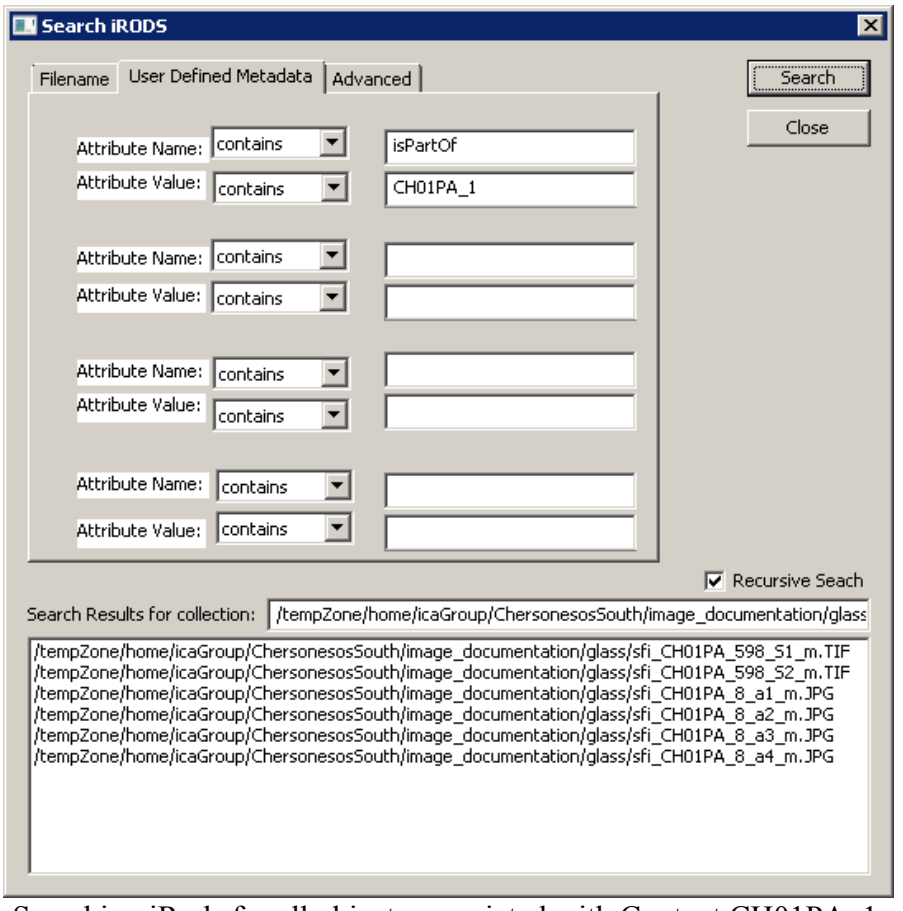

Figure 4. Searching iRods for all objects associated with Context CH01PA_1.

Conclusions

The creation of a record-keeping system from which metadata can be automatically extracted reduces the time spent in the labor-intensive, error-prone and often incomplete process of manual metadata entry. The resulting archive will preserve both the contextual relationships of the data and a record of their transformation as research progresses in a standardized, human-readable format. iRods provides the framework for automating metadata extraction, orchestrated by a Jython script utilizing common open source tools and metadata schema standards. This methodology of incorporating custom metadata extractors into iRods, particularly when combined with the wealth of information provided by FITS, will be easily extensible to other data collections. Currently, we are pursuing such work for collections in the domains of art photography and herbarium data. We are studying how to enhance the record-keeping system by building or leveraging existing file management GUIs to reduce the amount of manual naming and file categorization.

References

Esteva, M., Trelogan, J., Rabinovitz, A., Walling, D. & Pipkin, S. (2010). From the site to long-term preservation: A reflexive system to manage and archive digital archaeological data. In *Archiving 2010, Den Haag, The Netherlands: Final Program and Proceedings*. Springfield, VA: Society for Imaging Science and Technology.

Henderson, S. (2009) Personal document management strategies. In *Proceedings of CHINZ '09: The 10th International Conference NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction*. Auckland: New Zealand. doi:10.1145/1577782.1577795

Walling, D. & Esteva, M. (2010) iRods metadata extractor script (Version 1.1) [Software]. Retrieved May 5, 2011, from http://www.tacc.utexas.edu/~walling/irods_metadata_extractor.zip.