droid7

Table Of Contents

Formats to Identify

This page exists to document formats that you want to see DROID/PRONOM provide coverage for. Listing the format name, extension and any additional useful information will help The National Archives in getting the format into PRONOM.

An example template using the Bathymetry Attributed Grid format might be:

Format Name: Bathymetry Attributed Grid
Extension: *.bag
Website: http://marinemetadata.org/references/bag
Stakeholder: TNA, ADS
Comments: Any additional information you have that might be useful

Simply list the formats you require and put your institution next to the stakeholder field to help us prioritise our work. Additional information that might be useful will include links to specifications, sample files, resources already describing outline signatures for the format.

Format Name: ICC Color Profiles
Extension: *.icc
Website: http://www.color.org/icc_specs2.xalter
Stakeholder: TNA,
Comments: The National Archives does not currently have an approach to identify ICC color profiles. Despite a simple a well documented signature that can identify all previous and future versions of the format, it is complicated by its existence within 'container' formats. For example, within JPEG2000 an entire profile can be stored. DROID will identify the JP2 first, to identify the ICC profile as well DROID would need to report a multiple identification. Can DROID somehow report on both easily and unambiguously? Discussion topic here for anyone with any suggestions: http://droid7.wikispaces.com/message/view/home/49418268

Format Name:
Extension:
Website:
Stakeholder:
Comments:

Format Name:
Extension:
Website:
Stakeholder:
Comments:

Format Name:
Extension:
Website:
Stakeholder:
Comments:

Format Name:
Extension:
Website:
Stakeholder:
Comments:

Format Name:
Extension:
Website:
Stakeholder:
Comments:

Continue here...

Welcome to the DROID 7 requirements catalogue.

DROID 6 was released by The National Archives in March 2011. We are now looking forward to developing DROID 7. As part of the process we would like to follow we would like to consult with our stakeholders and gather a catalogue of requirements that we can understand and use to prioritise the development of the new utility.

We want the community to suggest requirements and proposals for how you would like to see DROID progress and improve.

We have created this wiki to act as a forum to help document this process. We invite you to participate by promoting any suggestions you think are important moving into the future. Existing requirements have come from feedback on mailing lists and informal chats with stakeholders about DROID and how it is used. They have also been developed from the joint Digial Preservation Coalition, The National Archives PRONOM and DROID User Consultation event. There is an opportunity using this process to connect with a wider comunity who were perhaps unable to make the day and those who still have more to input about the future of the tool.

Feel free to suggest new ideas, or identify your name or organisation as a stakeholder on an existing idea. Discussion in the discussion pages of this wiki or on the

//

requirements pages is also strongly encouraged.

The wiki structure we have adopted separates proposals into Description, Motivation, Solutions, Notes, Source and Declare Interest. Our wiki user guide explains a little more on how we have used this structure, here.

We would suggest that you create a log-in at wikispaces.com and request to become a member of the DROID 7 wiki. You can also log in with OpenID, including your Google or Yahoo accounts.

We are also collecting requirements via twitter and the hash tag #droid7. The feed can be found to the right of this page.

The wiki will be controlled at The National Archives, ideas will not be removed but may be edited or moved to match the style of other proposals or if they duplicate another proposal. Changes will be completely transparent with a full reason given for each edit by us, and all changes will be viewable in the history of pages. If there are any questions you can contact me at, pronom@nationalarchives.gsi.gov.uk.

Please note, the wiki at the moment represents an exhaustive list of requirements. We will use information from it to help outline a narrower and focussed set of requirements for DROID 7. We cannot promise all requirements will make it into DROID 7 but

they will act as input into DROID projects going into the future. Providing feedback and suggesting ideas is important. A suggestion, while it might not get implemented in DROID 7 could potentially be the first thing on the list for DROID 8 and so forth.

Thank you for using DROID and participating in this wiki, we look forward to your input in this new phase of development.

Functional Requirements Index:

01.	DROID Identifies multi-file format instances	02.	DROID provides reporting on unidentified file formats	03.	DROID provides a safe failure mode and resumes from point of failure	04.	DROID contains mechanisms for format validation	05.	DROID identifies attempts at data obfuscation (passwords / encryption / etc.)
06.	DROID has a PREMIS compatible XML output	07.	DROID provides machine readable XML output	08.	DROID allows submission of unidentified formats and incorrect identifications via the user interface	09.	DROID has a precise and 'fast' identification modes	10.	DROID provides a clearer, more public API
11.	DROI	12.	DROI	13.	DROI	14.	DROI	15.	DROI

	D reports more comprehensive file level data information (Created/Modified/Last Accessed)		D can identify formats using the Java native regex library		D can identify text based formats such as source code and scripting languages		D interface reports on percentage complete when scanning		D identifies more container formats
16.	DROID D provides a database free mode for identification of formats without the need for profile reports	17.	Where DROID D comes up with multiple identifications for a single file, it displays these in an ordered way (most likely option at top)	18.	DROID D allows for the upload of container signature files	19.	DROID D 7 uses a different open source license	20.	DROID D 7 implements the concept of Fuzzy signatures
21.	DROID D scans DB/EDRMS BLOBS	22.	The installation directory for DROID D 7 is user configurable	23.	DROID D 7 scans PST files and reports on the format	24.	DROID D 7 report breakdown reports in configurable units	25.	Drag and drop for files

				content of those files		(bytes/ kb/mb etc.)		
26.	DROID displays the signature file used per each profile tab	27.	DROID supports the testing of local signatures	28.	Signature files can be managed to be more modular and support local variations	29.	DROID supports the local ignoring or overwriting of specific signatures	30.	DROID remembers folder last scanned in folder structure
31.	DROID Command line is easier to use	32.	DROID identifies the creating application / environment of files	33.	DROID is able to load a list of files to identify from a file	34.	DROID Can generate a range of checksums for users	35.	Folder metadata via Active Directory hooks into / from DROID
36.	DROID better supports the development of new signatures	37.	Plug-ins can be added to DROID	38.	Ability to check against one fmt given from a partner through a local test	39.	Add Bzip2ArchiveContentIdentifier	40.	Having the possibility to get the File Format from a puid or a filename extension directly without "hacking"

								Droid internal data structures
41.	Having the possibility to exclude x-fmt puid from answers	42.	Mandatory match of signature and extension	43.	DROID flags files that are part of analysed container files	44.	requirement0044	45.	requirement0045
46.	requirement0046	47.	requirement0046	48.	requirement0048	49.	requirement0049	50.	requirement0050

Non-functional Requirements:

To capture some of the additional messages we have gathered via the #droid7 hash tag and the DROID consultation event (below), we have created a discussion to list non-functional requirements. Non-functional Requirements.

Formats to Identify:

The National Archives would like to collate a set of requirements for signatures you would like to see catalogued in PRONOM to be identified by DROID. Please follow the wiki link here to record formats of interest on a separate page. File formats to identify.

Digital Preservation Coalition: The Future of File Format Identification: PRONOM and DROID User Consultation

In preparation for the DROID 7 consultation, The National Archives in conjunction with the Digital Preservation Coalition held a one day event to discuss the two tools, DROID and PRONOM. The day included presentations covering DROID use cases and its benefits and limitations. The day culminated with a requirements discussion which has been used to seed the initial requirements on this wiki. Presentations from the speakers can be found below:

- Ross Spencer, When is a migration pathway not a migration pathway? — When is a migration pathway not a migration pathway?

PRONOM: Re-engineering a technical registry using Linked Open Data principles...

- Andrew Jackson, Linked People, Building a community around trustworthy data

AJackson-LinkedPeople-Nov2011.pdf

- Details
- Download
- 722 KB

- Dr David Tarrant, UHT Digital Preservation

DTarrant-UHTDigitalPreservation-Nov2011.pdf

- Details
- Download
- 550 KB

- Bill Roberts, Towards a format registry ecosystem

BRoberts-FormatRegistryEcosystem-Nov2011.pdf

- Details
- Download
- 317 KB

- Kate Pritchard, Mike Kaye, DEFRA experiences using DROID and using an IT provider

Defra-UseCase-ExperienceUsingDROID-Nov2011.pdf

- Details

- Download
- 122 KB

- Jenny Mitcham, Using DROID to create file level metadata – a case study from the Archaeology Data Service

JMitcham-DROIDCaseStudy-Nov2011.pdf

- Details
- Download
- 648 KB

- Johan van der Knijff, Evaluation of format identification tools

JvanDerKnijff-EvaluationOfIdentificationTools-Nov2011.pdf

- Details
- Download
- 885 KB

The video proceedings of the event are hosted on The National Archives media website.

Ability to check against one fmt given from a partner through a local test

DROID Identifies multi-file format instances

Description:

Formats which are composed of multiple files of differing formats are identified by DROID. GIS, CAD formats might be included in this example.

Motivation:

We have streams of information out there which are composed of multiple files which identify themselves as discrete formats. The digital preservation community requires a method of identifying them.

Solutions:

- Model this in PRONOM - rspencer_tna Jan 20, 2012
- Make it easier to use DROID export data to control this after a DROID workflow stage - rspencer_tna Jan 20, 2012

Notes:

- Some clearer definition of the types of formats users are coming across which match this instance would be useful here. - rspencer_tna Dec 19, 2011
- Is this something that should be handled by a tool after DROID has identified individual components? - rspencer_tna Dec 19, 2011
- An example is Shapefile: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf - rspencer_tna Dec 19, 2011
- ADS have lots of examples - tif world files, GML files, geophysics files etc - jen_mitcham
- We think there is scope here to look at the datamodel used to describe the file:puid relationship.- jaygattuso
- Perhaps a folder can be seen by DROID as a valid container, then the 'contains file' type language of the container signature model can be used to assert a complex file type with with children/sibling identity. - jaygattuso
- This identity could be signature based, or filename / filename_masks.- jaygattuso
- This notion should also allow the discrete files to have their own PUID (where applicable) and so a structured datamodel would be required that has a concept of a single file, and a single files 'place' in a larger structure - jaygattuso

I think this would involve a very big change to both DROID and PRONOM:
- It is true that container format recognition does something very similar (look for particular files embedded in a logical container file). Scanning a container file is predicated on first recognising the container format of the parent file (e.g. a docx file may be recognised if the containing file is a zip file). The final identification is applied to the containing file. On what object would we make an identification? Would we flag the containing folder as being of the mult-file format?

- Treating file system folders as potential multi-file format containers would involve scanning every single folder to see if certain files of particular formats were present. This would almost certainly be very slow.
- How would you deal with folders which (for example) contained shape files, but also contained other things?
- PRONOM and DROID's internal code, GUI, export formats and reporting are all predicated on a file-->formats identification model. Changing this would be a lot of work.
I'm not saying this is impossible... but given the large scope of change required, I suspect that multi-file format identification may be something best left to tools that subsequently process the DROID output. - MattPalmer Jan 24, 2012

- totally agree Matt, but at some point we should have the conversation... this seemed like a good as time as any... :) - JayGattuso Jan 25, 2012

- May be simpler to implement this as a file-level identifications, but adding a relationship result that indicates that a file of one format should be accompanied by another file with some specific format/name. This avoids having to make an identifier for the set. - anjacks0n
- Of course, JHOVE2's Aggrefier does this, and so we could reuse their code or defer to them. - anjacks0n

Source:

DROID Consultation Event

Declare Interest:
ADS
NLNZ
BL
ExLibris

NLA - JonathanMcCabe1 **Jan 26, 2012**

DROID provides reporting on unidentified file formats

Description:

There is clearer and more useful reporting of unidentified file formats in DROID.

Motivation:

The unidentified formats are the ones that we are most interested in the majority of the time. Given an accurate identification engine we accept the result of a positive identification but what do we do with the unidentified files? How can we begin to understand the quantities and sizes of this potential problem in DROID.

Solutions:

Notes:
-Would it be possible to output a structured object that contained (e.g.) the first and last nKB of an unknown file, an extension (if found) and any O/S MD e.g. date created, date modified, last accessed, size etc. This packet of data could be submitted to a knowledge base for investigation by any motivated party.
- jaygattuso

Source:

DROID Consultation Event

Declare Interest:

ETH

DROID provides a safe failure mode and resumes from point of failure

Description:

DROID has awareness of what point it has reached in the scan of a file system to enable it to rescan from that point should it fall over for whatever reason.

Motivation:

Currently, if DROID falls over mid-scan it doesn't know how or where to resume scanning. This means scanning a potentially large number of files all over again. Scans which take multiple days require the same time again.

Solutions:
In the absence of the ability to resume in the face of DROID itself crashing, a simple (but not entirely satisfactory) work-around is to split up your file scanning into more than one profile and scan less information in each. This limits the damage of a crash in any one scan, and doesn't stop you aggregating the results for reporting or exporting. - MattPalmer Jan 24, 2012

Notes:
DROID 6 already has a partial safe failure mode, which is admittedly not perfect. If DROID fails to read from the file system because the file system has gone offline, it should automatically enter Pause mode, allowing you to save your work so far, and to restart once the file system is back on-line. However, I do agree that the ability to restart a profile when DROID itself has crashed would be very useful. There are a number of issues in doing this. One is that it is currently quite hard to restart processing in the middle of an archive file (e.g. if you are in a zip file). I believe (but am not 100% sure) in the current system, if you save a paused profile, it discards any partial processing of archive files (meaning they have to be entirely rescanned on restart). However, this is probably a price worth paying for such resilience. Other issues are that DROID relies on information being saved properly in the profile.xml file. I guess it should be possible to reconstruct where processing was up to by examining the content of the database instead, although this may be trickier given that DROID processes files multi-threaded, so I'm not entirely sure there is such a thing as a single folder it is working on at any given moment in time. The final issue is that processing of DROID files is done in a temporary file area - there is currently no way of telling DROID to pick up a partial profile from the temporary area instead of loading it from a saved file - but this shouldn't be too hard. - MattPalmer Jan 24, 2012

Source:

DROID Consultation Event

Declare Interest:

NLA - ETH JonathanMcCabe1 **Jan 26, 2012 We are very interested in this**

DROID contains mechanisms for format validation

Description:

DROID can identify a file and validate whether it conforms to the expected specification.

Motivation:

It can be argued that a format is only what it purports to be if it matches the specification to which it was supposed to be built from. DROID might be a place to add format validation to provide a concrete answer as to the identity of a particular format.

Solutions:

Notes:

- Is this scope creep? - rspencer_tna Dec 19, 2011
- This seems like a huge step up from the current characterization that DROID undertakes, it would be useful, but seems like a large overhead to manage the validation functions required, especially given the broad coverage of the PRONOM registry. Maybe it would be possible to make a format ID extensible via a plugged in validation tool - the user is required to create / manage the tool, by DROID supports the linking of a PUID to a validation tool via disclosed API? - jaygattuso
- I think this is scope creep. DROID's entire model is one of probabilistic file format characterisation of thousands of formats, based on short signatures for each. Validation requires building in knowledge of each file format to a highly detailed level. It's not even clear what constitutes a valid file - there are many, many HTML or PDF files which are not strictly valid, but no-one would dispute that they are HTML or PDF files! Or at least, that they should be treated as such for preservation purposes! Doesn't JHOVE deal with format validation? - MattPalmer Jan 24, 2012

- indeed, and for linking the two together there is always FITS which is the direction this thread is kind of heading.... - JayGattuso Jan 25, 2012

- I agree, this is best deferred to other tools. - anjacks0n

Source:

DROID Consultation Event

Declare Interest:

DROID identifies attempts at data obfuscation (passwords / encryption / etc.)

Description:

Formats which are password protected or exhibit other forms of data obfuscation such as encryption are identified and flagged by DROID.

Motivation:

These formats present a risk to collections and so need to be identified so that they can be handled within any preservation work flows.

Solutions:

Notes:
- Some research was done on this during the DROID 5 and 6 development. We need to distinguish between formats with a naive password protection scheme and those which actually encrypt the content.
- A naive protection scheme doesn't actually obfuscate the content at all, and simply relies on the opening application "respecting" the password. Clearly, if there are open source or other applications which ignore this password, then there is no particular preservation issue.
- For files in which the content is encrypted, there can be an identification issue. If there is insufficient unencrypted content in the format to allow the format itself to be identified (except possibly by the file extension), then DROID can't even figure out it's an encrypted format to begin with, as no signature can match it. If the format has some kind of structure which flags that the content is encrypted, then DROID could recognise those structures.
- One option which was explored was to calculate the Shannon entropy of a portion of each file. This gives us a measure of how "dense" the information in the file is. Highly compressed or encrypted files have a high entropy score. While this doesn't tell you a file is encrypted, you could filter the results, looking for files with no identifications, which also have a high Shannon entropy. These files are candidates for being encrypted, even if there is no other identifying structure in them. Some (probably not very optimal) sample code to calculate the entropy is in the attached file.
- Note: using the first parts of a file to identify whether the content is encrypted doesn't work very well, as the file headers for encrypted formats are often not encrypted (but may not offer a structure that DROID is capable of recognising, flagging that the content is encrypted). Calculating the shannon entropy for the entire byte stream of a file would be very slow. So there are some trade-offs between performance and accuracy here, as usual. - MattPalmer Jan 24, 2012

shannon-entropy.txt

- Details
- Download
- 2 KB

- Reporting the Shannon entropy would be of interest, but as it cannot really tell between compression and encryption, I'm not sure we should expect too much from it. - anjack0n. BTW, I have a Java-based SSDeep fuzzy hash implementation, which might be interesting if you are into that kind of thing. - anjacks0n
- Well, you are right it can't distinguish that. But you can combine the information from Shannon with other info to drill down on files of interest. In particular, most common compression formats will already by identified as such, as signatures for them are well known and developed. Secondly, encrypted formats usually retain their normal file extensions (for once, hurrah for file extensions!), so, for example, you could look for files with an extension mismatch warning that have a high Shannon entropy.... this would give you a fair list of files which were potentially encrypted (or had some previously unknown compression option!). And it's just cool seeing the information density of different file formats anyway (well, for some of us, I guess...)
- Very interested in the SSDeep fuzzy hash. In fact, duplicate (and near-duplicate) file detection was on the original agenda for DROID 6, but we had no time or resource to pursue it... - MattPalmer Feb 8, 2012
- Well, the code is here, and you're welcome to use it. At some point I'll repackage it into its own project, but for now it's mixed in with the SCAPE code. - anjacks0n

Source:

DROID Consultation Event

Declare Interest:
ExLibris

DROID has a PREMIS compatible XML output

Description:

DROID outputs a report of identification compatible with PREMIS.

Motivation:

This makes it easier to use DROID in systems already using PREMIS metadata.

Solutions:

Notes:

- How widely used is PREMIS amongst institutions running DROID? - rspencer_tna Dec 19, 2011
- Do we need PREMIS and additional metadata formats? - rspencer_tna Dec 19, 2011
- I'm not sure this is possible - are the data dictionaries/PREMIS events standardised enough that you'd know how to format it? - anjacks0n.

Source:

DROID Consultation Event

Declare Interest:

NLNZ

NLA - JonathanMcCabe1 **Jan 26, 2012**

DROID provides machine readable XML output

Description:

DROID provides an export that is machine readable for more programmatic use of the DROID output.

Motivation:

To make DROID exports easier to use by IT systems.

Solutions:

- Initial ideas for DROID 5.0/6.0 suggested the use of a triple store back end. Could SPARQL provide the export functionality we need?
- Not sure if a triple store database would scale to the type of filtering and reporting required. In any case, regardless of the underlying database technology used in DROID, getting DROID to write out its profiles in an XML format is not hard. The current DROID architecture would make this fairly simple to achieve using JAXB or some other standard Java XML technology.
- Once you have DROID data exported in a DROID-specific XML format, it may be possible to transform it using XSLT to any other reasonably compatible XML format such as PREMIS. This is exactly how the PLANETS XML reports are generated in the existing DROID 6. - MattPalmer Jan 24, 2012 Of course, the reports are a lot shorter than a full export of millions of profiled files, so this may not be viable.
- I'd actually prefer plain old CSV. You can map it to whatever you like, it's easy to use with command-line tools, and when it gets really big you can query it via Hadoop! - anjacks0n
- Well, CSV is already there, in two different flavours! But I guess the point is, don't lose it if you add XML! - MattPalmer Feb 8, 2012
- Ah, of course it does! Sorry about that. - anjacks0n

Notes:

- Is the requirement for http://droid7.wikispaces.com/requirement0006 actually machine readable XML and not PREMIS specifically (something that can be transformed and adapted as needed) - rspencer_tna Dec 19, 2011

Source:

DROID Consultation Event

Declare Interest:

NLA - JonathanMcCabe1 **Jan 26, 2012**

DROID allows submission of unidentified formats and incorrect identifications via the user interface

Description:

DROID allows users to submit information via the user interface of instances where formats have not been identified or have inaccurately been identified.

Motivation:

To provide The National Archives with more useful information about gaps in their coverage.

Solutions:

Notes:

- With a clear licensing statement can this solution go as far as allowing DROID to submit the file in question? - rspencer_tna Dec 19, 2011
- Is security compromised if we do this? - rspencer_tna Dec 19, 2011
- Should we export a zip containing data relevant to the submission and ask the user to manually send it to TNA? - rspencer_tna Dec 19, 2011
- This would be great as a function - export known data (extension, OS MD, header/foooter hex etc) as a structured object, and have a supporting repository that can be searched by 3rd parties - the value would be in externalizing the repository so we can all look for matches etc in there. - jaygattuso
- Submitting content like this does open all sorts of security and legal issues on the file content. Is there a good channel already open to submit files for signature development to the National Archives already? Not sure this is really a software issue. - MattPalmer Jan 24, 2012
- I would posit the software aspect would be the packetisation of unknown stuff in a structured way. If this is suitably harvested / made accessible in a much more consistent way. Scope creep? yupe, but a cool thing to have no doubt (hence the NLNZ interest caveat!) - JayGattuso Jan 25, 2012

Source:

DROID Consultation Event

Declare Interest:

Tentatively NLNZ - but as an external service, not as part of DROID....
Tentatively BL - external service preferred.

NLA - JonathanMcCabe1 **Jan 26, 2012 we agree with NLNZ**

DROID has a precise and 'fast' identification modes

Description:

DROID provides modes of identification that focus on the precision of accuracy and modes that focus on speed of identification.

Motivation:

This is to allow reccys of collections in quicker time. One might then run a complete scan to get the detailed data from DROID required for preservation.

Solutions:

Notes:

- What methods exist that promote the speed element of this requirement? - rspencer_tna Dec 19, 2011

- maybe the creation of a smaller subset sig-file? this would only work if the user is expecting a subset of files being submitted. - jaygattuso

- DROID already has some options to trade off accuracy against speed. The simplest way to increase the speed of DROID is to limit the number of signatures it processes, or use cut-down versions. I like the idea of a signature manager application, which would allow the creation and testing of new signatures, and the management of existing signatures. - MattPalmer Jan 24, 2012

- I think this sounds like a lot of work setting up 'coarse' and 'fine-grained' signature sets. Might be better to optimise elsewhere, e.g. fast process all 'fixed string starting at zero' signatures by using a binary-tree/hash-table of the first few bytes in the file so that only probably contenders are trialled. - anjacks0n
- There is a functioning advanced Trie structure matcher in the upcoming byteseek 1.3, which will work with the same sorts of sequences used in DROID (including those with character classes). The idea would be to do a multi-sequence match on all signatures starting at the same fixed point, and only bother to continue matching those that passed this filter. For simple signatures with only a fixed sequence, this would be all the matching required. This would allow DROID to scale largely independantly of the number of signatures, doing most matching in a single pass. What would be left would be any sub-sequences that followed a Trie match. Many of these are repetitive, and could be handled by a multi-sequence search (e.g. Wu-Manber). As always, the goal would be to avoid doing the naive thing of running each signature individually against the same byte stream again and again. I don't think it could ever get to a single pass over the stream, but the number of passes and effort can certainly be reduced significantly.- MattPalmer Feb 8, 2012

Source:

DROID Consultation Event

Declare Interest:

"signature manager application" - NLNZ
"signature manager application" - ETH
"signature manager application" - BL
"signature manager application" - ExLibris

DROID provides a clearer, more public API

Description:

DROID opens up the API and documents it in a more detail to allow better programmatic integration of the tool in IT systems.

Motivation:

This will give programmers more choice when integrating DROID in IT systems.

Solutions:

Notes:

- Does this requirement clearly distinguish between what TNA has called the API in the command line interface and more low level access methods? - rspencer_tna Dec 19, 2011
- While I think the command line interface facilitates scripting DROID into wider systems, there is a big hole in the programmatic side. There is no clearly defined programmatic API in DROID. While it's a very modular system, it's really not clear how to approach the code. I think the commonest requirement is to be able to embed the core identification ability of DROID into other software (probably losing the GUI, command line, reporting, exporting, filtering, etc.). At present, it's really not clear which bits of DROID are required for this. Better documentation would certainly help, but centralising the identification capability behind some clearly demarcated interfaces would help a lot too. - MattPalmer Jan 24, 2012
- The code may be modular, but attempting to invoke the SubmissionGateway programmatically revealed that the modules are highly interdependent. Spring calls Java which calls Spring, and the 'profile' and JPA stuff is always pulled in whether you like it or not. I gave up trying to pick the Spring apart and have never managed to invoke the full DROID identification system programmatically. The binary signature identifier was invokable, but using it in combination with the container identifier was beyond my ken. I really just want a JAR I can call a method in and pass it a File or InputStream and it will do the rest. No Spring, no JPA, just a result please. - anjacks0n.
- Yes, the submission gateway is far too involved with identification. It was supposed to be a multi-threaded work dispatcher, not an identification engine. Splitting out identification entirely from all the rest of DROID would be a good thing to do. - MattPalmer Feb 8, 2012
- In Rosetta we implemented DROID 6 by calling the appropriate java classes but it was not clear at all what is the appropriate way doing so. We agree that a clear simple callable APIs are required.nir.sherwinter (ExLibris)

Source:

DROID Consultation Event

Declare Interest:

NLA - JonathanMcCabe1 **Jan 26, 2012**

DROID reports more comprehensive file level data information (Created/Modified/Last Accessed)

Description:

DROID reports on Create Date, Modified Date and Last Accessed dates on all digital objects accessed.

Motivation:

This additional information will be beneficial to archives and memory institutions in terms of understanding provenance and driving other preservation decisions.

Solutions:

- Does Java 1.7 provide this level of access to file level metadata?
- Yes, Java 7 provides better access to file system metadata. However, Java 7 is not yet widely deployed in many organisations. Would you want to maintain a Java 6 and a Java 7 version? - MattPalmer Jan 24, 2012

Notes:

Source:

DROID Consultation Event

Declare Interest:

DROID can identify formats using the Java native regex library

Description:

DROID provides the additional identification mechanism of standard regular expressions via the native Java regex library.

Motivation:

To increase the number of options available to those who want to create signatures for identifying formats.

Solutions:

Notes:

- Is the Java regex library the requirement or do better libraries exist for this purpose? - rspencer_tna Dec 19, 2011
-- The latest fancy RegEx engine appears to be http://www.brics.dk/automaton/ which is being ported into Lucene 4. Those FSM algorithms apparently have much better overall performance that the usual RegEx implementations. - anjacks0n
-- Deterministic FSM's are faster than their non-deterministic brethren to be sure, but none are very good at searching across large areas of 'text' - they are good at matching closely arranged patterns of information. You want sub-linear search algorithms for that (e.g. Boyer Moore Horspool). - MattPalmer Feb 7, 2012
- We would support moving the signatures into native regex format, or maintaining a regex version alongside the 'DROIDexp' (where possible) - the specific implementation is less important - jaygattuso

- I would argue against this requirement as it is currently phrased, for several reasons. A significant amount of DROID's performance comes from its ability to use advanced searching algorithms (e.g. Boyer Moore Horpsool), which skip over bits of a file which can't match. Java regular expressions cannot do this - performance would drop off a cliff.
-- Note: it has been pointed out that Java's regular expressions also use Boyer Moore internally! - MattPalmer Feb 7, 2012
-- Further note: it transpires that the use of Boyer moore in java regular expressions is ONLY enabled if you compile the expression as a case insensitive literal string - in other words, only if you don't use it as a regular expression! The Boyer Moore Horspool searching in DROID is already more advanced than this, as it can handle case sensitive (ASCII) and character classes (actually byte sets) in its search, through an extension of the normal BMH algorithm. So, I'm afraid we don't get super fast regular expression searching by default in Java. - MattPalmer Feb 8, 2012
- I do sympathize with the motivation, but I think this can be addressed by other means. In particular I would like to see a dedicated (or built in) signature manager which allows the creation and testing of signatures. The DROID signature XML is particularly hard to interpret, but the underlying syntax supported by DROID 6 is actually fairly close to normal regular expressions (albeit, byte oriented, rather than text oriented), so the learning curve here would actually be quite small. - MattPalmer Jan 24, 2012

- Matts comment noted, it seems the value is in PRONOM, in an accessible standardised regex representation of the signature patterns. Perhaps the this could be addressed by adding a datafield to the PRONOM record rather than changing the underlying method used in DROID?- JayGattuso Jan 25, 2012
- I like Jay's idea here. The value here is in the signatures and the understanding and re-use they facilitate, not the underlying code used to implement them. The DROID implementation has some specific features which work well for the kind of things DROID needs to do. But there is no reason that the signatures themselves could not be represented in PRONOM in a more standardised form. - MattPalmer Feb 7, 2012

- There is another objection to directly using the Java regex library: it only works on valid character sequences, not byte sequences. I guess it might just be possible to read the bytes from a file into a char array rather than a byte array (although this would be at least twice as large as the corresponding byte array, as a char is an unsigned 16-bit value, whereas byte is a signed 8-bit value, and would have another performance hit, as all reads would have to be copied from a byte array into a char array).
- The DROID XML is quite horrid, but a change in DROID 6 means that all the and elements are actually entirely ignored (these values are calculated by the software). It is entirely legal to create a DROID binary signature file for DROID 6 omitting all these elements. This makes writing binary signature files (by hand) much more doable (I've done it - although it's not very pleasant still!). In fact, you can use the syntax you find in the container signature files in the binary signature files too (e.g. you can specify ASCII strings using single quotes, etc.). It then starts to look much more like a set of regular expressions (although the min and max offsets are still a bit offputting).
- At the very least (in the absence of a GUI or command-line tool to parse the full regular expression syntax and transform it into DROID binary signature XML), it would be good to have documentation describing how to write DROID 6+ binary signature files, with the full syntax available in it - MattPalmer Jan 27, 2012
- A more serious problem with most regular expression libraries is that they cannot match expressions which traverse more than one buffer, whereas DROID has been engineered to allow matching across an unlimited amount of a stream or file - but without requiring the entire stream or file to be loaded until necessary. Using a standard regex library forces you to pick a buffer size, always load the entire buffer (e.g. 64Kb), and never match anything bigger than the buffer. - MattPalmer Feb 7, 2012
-- This is not true. You can implement a CharSequence and do the buffering under the hood. The CharSequence serves the same functionality as your ByteReader, if I'm remembering your class's name correctly. - anjacks0n.
-- Furthermore, if only BOF signatures are needed, this point is entirely moot. Tika and friends use the first 8KB - anjacks0n
-- I stand corrected! And with that I think many of my technical objections to using Java's regular expressions melt away! Real-world performance comparisons would be nice to have. - MattPalmer Feb 7, 2012
-- Well, some more analysis also shows Java's regex library seems to need to know the total length of the CharSequence - which is problematic for streams, forcing you to read in the entire stream to determine its length - or at least as much of it as you would want to match against ahead of time. Now, DROID currently needs to know the stream length as well (particularly if it matches EOF signatures), so this may be a moot point. However, the upcoming 1.3 release of the byteseek library DROID uses will remove this requirement, potentially allowing for more performant and stream friendly processing. The syntax of the regular expressions used for the signatures is a separate issue, in that the Java regular expression syntax could be used (or some sub-set thereof) without regard to the underlying implementation - MattPalmer Feb 7, 2012

- There are things that DROIDexp can do but RegEx cannot, but it seems that none of the signatures need this extra functionality. Therefore, the arguments against RegEx appear to boil down to performance. The performance gain of DROIDs implementation is unclear, given that Java uses Boyer-Moore under the hood. SCAPE is attempting to probe some of these aspects with large-scale tests. - anjacks0n.
- But the deeper point is about sustainability. We are expending time and money looking after code that supports a domain-specific matching language. Not only can we save money by using an implementation someone else maintains, but by using a more standard language, we can widen the signature community. You can hire people who know RegEx, or send them on a course, or tell them to read a book about it. I can't say the same about DROIDexp. For me, the positives have to be pretty huge to outweight this negative. - anjacks0n.
- I'm much more sold on the sustainability of a standardised pattern matching language, rather than the code expense. The actual effort involved in writing the DROID software was only about 5% focussed on the underlying matching code used, which underwent some refactoring in DROID 6, but is still fairly similar in spirit to the code which existed in DROID 1! The vast majority of development effort is in the GUI, command-line, reporting, filtering, exporting, signature management, etc. However, making it easier to develop, understand and share signatures would be a huge win - which is reflected in other parts of this wiki by the interest shown in opening out signature development and managment by DROID and PRONOM users. - MattPalmer Feb 7, 2012
- I agree, it's not really about the matching engine, but about the syntax and making it easier to extend DROID. I would still prefer to see us save that 5% and spend the effort elsewhere, but I accept that it's not that much code overall. The Fido codebase shows how to automatically map DROIDexp to RegEx, so perhaps it would be simpler to look at providing the reverse mapping? That does create more code to maintain, but avoids significant changes elsewhere in the system. - anjacks0n
- Having said all of that, if the Java regular expression engine is fit for purpose, then it would probably be better to use it than a proprietary solution (inc. my own byteseek library). Right now, my only real concerns would be potential performance issues and the stream-unfriendliness of the Java reg exes needing to know the length of the sequence they will be matched against ahead of time (which the current release of byteseek also requires, but the next release doesn't). - MattPalmer Feb 8, 2012

Source:

DROID Consultation Event

Declare Interest:

tentatively NLNZ

DROID can identify text based formats such as source code and scripting languages

Description:

DROID is capable of identifying text based formats such as source code, structured text, etc. and assigning a PUID to that identification.

Motivation:

There are a large number of these formats out there which we just can't identify using signatures. An alternative mechanism is required to allow us to identify these objects on ingest.

Solutions:

- OhCount
- CLOC
- Other code counting mechanisms?

Notes:
- this would be a very useful capability, I suggest a 2nd parsing of objects ID'ed as text, looking for formatting (either inside the text, or the encoding) patterns that are unambitious. We have some examples of text that should be differentiable in this way. - jaygattuso

- Some work was done on text heuristics in DROID 6 development, but there wasn't time to flesh it out into a finished solution. There are several issues with text file identification which need to be addressed.
1) The first issue is to (fairly) reliably distinguish between binary formats and text formats in the first place, as many of the text heuristics which could be applied would be quite slow if applied to all files all of the time. This turns out to be quite tricky, but I did come up with some heuristics which were surprisingly reliable when run over thousands of mixed text and binary files, including text files in most common encodings, including ASCII, Extended ASCII code pages, Unicode (UTF-8, UTF-16, UTF-32), and Chinese and Japanese encodings. The essence of the heuristic is to try to rule out binary files, until you are left with a probable text identification (and an encoding):
a) Scan only a fixed amount of the file (e.g. 1024 bytes)
b) Count the frequency of each byte in that buffer.
c) Count the occurence of consecutive "bad text" characters which normally do not appear in text files no matter what encoding is used (or if they do, they do not appear consecutively). - e.g. values less than 0x20 (but not new line, carriage return, tab, etc.). 0x7F. If you see consecutive "bad" characters, it is almost certainly not a text file. Binary files frequently have runs of zeroes, or other low value bytes.
d) What remains are files which could be text files, but can still be binary files. Now calculate the shannon entropy of the data, using the byte value frequencies previously calculated. Text files do not usually have a very high entropy (like media, compressed or encrypted files), as they contain regularities (the text!), and usually do not use all available byte values. Care is needed here - english text files encoded in ASCII have an entropy of around 3-4 bits per byte - but Japanese and Chinese encodings, or the unicode encodings can have entropies as high as 6.5. Media, compressed or encrypted files will normally have very high entropies - i.e. 7.5-8 bits per byte - but since they generally also contain all

possible byte values scattered apparently randomly, then it is quite probable you will not rule them out by seeing consective bad chars in the preceding step. The entropy test allows you to rule out media, compressed or encrypted files.
e) Discover the encoding of the probable text file. There is a good library for this: International Components for Unicode
This can still fail to detect an encoding, in which case regard the file as binary.
2) Once you can open a file as a text file with the correct encoding, it's good to able to figure out what sort of file it is. There are several sorts of text file, which we can broadly characterize as structured text (e.g. CSV) and unstructured files (e.g. programming languages, HTML, RTF, basic text in some language). I didn't do much work on recognising structured text files, but to recognise particular languages, the Aho-Corasick algorithm allows searching for thousands or (even millions) of words in O(n) linear time with respect to the amount of text scanned (not the number of words being searched for). There is a reasonable implementation in Java here.
3) You have to define what sorts of heuristics recognise different file formats, and define a signature format to describe them. An early stab at this was made, but it's incomplete work I did in my own time. Maybe there's some use for it yet... - MattPalmer Jan 24, 2012

- If you want to revisit this idea, we would be interested in coming along with you - JayGattuso Jan 25, 2012

- I still have some source code for text / binary distinguishing, and encoding detection. I also have some very preliminary code for keyword recognition using Aho Corasick, and some text files containing keywords for all the different RTF formats, xml and html. I would be happy to supply them to the DROID 7 project if there is any interest in pursuing it, or to discuss it further... - MattPalmer Jan 26, 2012

- If making a full 'clean room' implementation is too much, we could look at reusing some existing work. As well as the clever parser in ohcount (which is written in Ragel and could be compiled to Java to get things started) there are various SLOC tools we could use (I've collected some here), or we could perhaps use a syntax highlighting package instead of a lines-of-code parser (see e.g. here). Of the options I'm aware of, the quickest win for DROID would be running GitHub's Linguist via JRuby, but that would be difficult to integrate with PRONOM. - anjacks0n.
- Scratch that: GitHub Linguist relies largely upon file extension, with some pretty vague guesswork applied when the extension is missing. - anjacks0n
- NOTE that this documents how browsers do text detection - might be useful: http://mimesniff.spec.whatwg.org/#text-or-binary - anjacks0n
- Also, there is some effort and interest knocking about in improving text ID. If any half-decent prototype code could be made generally available, perhaps as a dedicated branch on GitHub, then we might be able to get others interesting in working on it. - anjacks0n

-I don't have time to understand GitHub, set up accounts, etc. But I do have my original TextEncodingDetector Java code, which can distinguish binary data from text enodings reasonably well, and give you the encoding if it believes it is text. The file is now uploaded and accessible from this page, released under a BSD license. It uses ICU4J to do the actual text encoding. The additional step is in heuristically ruling out binary data before we try to use that component. ICU4J doesn't do very well when arbitrary binary data is thrown at it - you need to be fairly sure its text before trying to use it. So this code adds binary-distinguishing heuristics. It has been tested over thousands of mixed text and binary files, using all common (and some uncommon) text encodings, mixed with binary files of various sorts. -

MattPalmer Oct 14, 2012

Source:

DROID Consultation Event

Declare Interest:
ADS - jen_mitcham
NLNZ
BL

NLA - JonathanMcCabe1 **Jan 26, 2012**

ETH - 2nd run idea for text based formats

TextEncodingDetector.java

- Details
- Download
- 20 KB

DROID interface reports on percentage complete when scaning

Description:

DROID has a percentage complete bar when scanning objects to provide users with more feedback.

Motivation:

To provide more detailed feedback about what point DROID has reached for any given scan.

Solutions:

Notes:

- DROID already has a progress bar, which admittedly isn't perfect. It counts the number of files to be scanned, and once this is done, progress is shown in terms of how many files are left to identify. However, scanning zip or tar archives is hard to estimate ahead of time (as you don't know you've got a zip or tar until you actually recognize it, so you can"t know how many files in total will be scanned). Various options were discussed during development (such as modifying the file count as archives are encountered - but this could have the counter-intuitive result of making the progress bar move backwards!). Does anyone have any suggestions for how progress reporting could be improved? - MattPalmer Jan 24, 2012

- I have heard that this is a very very difficult thing to get 100% right for the reasons Matt suggests and many more, some documentation about this issue from Microsoft is available here.
- EuanCochrane Jan 24, 2012

Source:

DROID Consultation Event

Declare Interest:

DROID identifies more container formats

Description:

DROID can identify more container formats and their contents.

Motivation:

To increase DROID coverage and give users a bit more information on files which can potentially contain more preservation risks.

Solutions:

- Apache Commons Compress

Notes:

- Scope creep? Should DROID be identifying the contents of containers or simply the container? - rspencer_tna Dec 19, 2011
- There is something in the notion of an intentional container (where my intent is to use the container as the format object, e.g. a docx, which is ostensibly pk zip wrapped xml) and a 'functional' container (where my intent is to use the function the container files gives, and not to infer anything about format type - e.g. a pk zip containing a bunch of jpeg files. - jaygattuso
- The earlier builds of DROID 6 not only recognized DOCX files, but also went ahead and recognized the files inside them. Beta testers did not like this at all. As you say, a DOCX is an "intentional" container, whereas ZIP is merely an archive format for the files inside them. But I agree, it should identify more container formats! Signatures for container formats are fairly simple - they just lists which files should be located and (optionally) whether they have one or more binary signatures attached to them. - MattPalmer Jan 24, 2012
- in my limited experience container sigs are astonishingly simple to write, and very powerful - so perhaps the value is in other sig management/devo tools as per other requirements. - JayGattuso Jan 25, 2012
- Seems the scope needs clarifying. If this means 'more container signatures please', then that's fine. If 'identifying their contents' means more than this, then I suspect we are asking too much. - anjacks0n
- Please include the ability to identify the contents of .iso files. We have a few in our collections and I can anticipate many more to come. - DClipsham Apr 26, 2012
- This is a good idea, but being able to parse ISO files into their constituent files is not a container format issue (you don't identify the ISO file as something else depending on what you find inside it). Adding a new Archive type handler for the ISO format would be the way to achieve this (in the same way that zip, gzip and tar are currently processed, or that PST files might be) - MattPalmer Apr 27, 2012

Source:

DROID Consultation Event

Declare Interest:

NLNZ

NLA - JonathanMcCabe1 **Jan 26, 2012**

ExLibris

DROID provides a database free mode for identification of formats without the need for profile reports

Description:

DROID can be run without the initialisation of the Apache Derby database.

Motivation:

This is for cases when the complexity of DROIDs reporting mechanisms are not needed. The tool focuses simply on identification and the export of those results can be analysed by tools other than DROID.

Solutions:
- Defining a console output module itself isn't really that much work. The architecture is quite modular - so results could simply be written to a new implementation of the results interface. There is probably more work involved in just modifying the aspects of the DROID GUI application which expect to be able to navigate, filter, report and export the data profiled and providing some sort of configuration to allow these sorts of profiles to be run. Maybe this could be a command-line only option if that is the case?

Notes:

Source:

DROID Consultation Event

Declare Interest:

Suggestion: output in csv format as non buffered stdout stream. This way it can get redirected to other processes/scripts.
Maurice de Rooij, National Archives of the Netherlands.
BL (prefer a command-line varient that streams the results rather than attempts to persist them in a profile).

Where DROID comes up with multiple identifications for a single file, it displays these in an ordered way (most likely option at top)

Description:

Sometimes DROID gives multiple possible identifications for a file. Could there be some rules in place to suggest which is most plausible identification?

Motivation:

We store metadata produced from DROID in a database. Currently only allow one identification per file in our data structure. It means we are not necessarily storing the right identification for a file. We may need to change our data structure instead!

Solutions:

- We could permit 'parent' or 'superclass' format records, so that ambiguous formats can be collected together under a single record (as suggested for TIFF by Jay). - anjacks0n

Notes:

Sorry - this may be a big can of worms!

Source:

Archaeology Data Service - jen_mitcham

Declare Interest:

DROID allows for the upload of container signature files

Description:

Currently DROID 'upload signature file' only applies to standard signature files. 'upload signature file' will also allow the upload of container signature files.

Motivation:

DROID should allow for the upload of all required types of signature file.

Solutions:

Notes:

This is something which is required especially as The National Archives wishes to see greater community participation in the development of signatures.

- Definitely should be done. It was an oversight that it wasn't in the original development. - MattPalmer Jan 24, 2012

Source:

- rspencer_tna Jan 20, 2012

Declare Interest:

- **rspencer_tna Jan 20, 2012**

NLNZ

NLA - JonathanMcCabe1 **Jan 26, 2012**

ETH

DROID 7 uses a different open source license

Description:

DROID 7 uses a different license to previous versions.

Motivation:

Currently DROID is licensed under Open BSD. We need to discover if there is an alternative that the community can recommend other than this that keeps DROID open to the community but allows it to take advantage of commercial uses of DROID.

Solutions:

Notes:

- If you want the license to force anyone who integrates the DROID code to submit any useful modifications they make back to the community, you're looking at copy-left style licenses like GPL. However, strong copy-left licenses preclude integration of the code into closed-source code (effectively preventing it from being used in such applications). You could look at the LGPL (which permits the unmodified inclusion of the code into closed-source products, but require any modifications to the code itself to be released). However, this is targeted at code libraries, not stand-alone applications. However, I'm really not sure why this is an issue. Do we think there are commercial entities making useful and secret modifications to the DROID code which we would want to re-integrate? - MattPalmer Jan 24, 2012
- perhaps the motivation for needs some clarification. I understood this requirement in a very different way, more as a method of monetizing the resultant IP (when used in commercial products). I would welcome any clarifications to this requirement - JayGattuso Jan 25, 2012
- I also request clarification of this licence. The BSD licence presents no barrier to commercialisation (and indeed, I know DROID is embedded in a number of commercial tools). You could switch to Apache V2.0 but I'm not sure you'd gain much. Switching to copyleft seems to be in contradiction with your requirement. - anjacks0n.
- I would, however, prefer it if you could make it clear what the licensing conditions of the format records and signature files are, e.g. by embedded the licence details in a comment in each file. - anjacks0n

Source:

- rspencer_tna Jan 20, 2012

Declare Interest:

- **rspencer_tna Jan 20, 2012**

DROID 7 implements the concept of Fuzzy signatures

Description:

DROID can report accuracy of non-100%-identifications by reporting on the percentage of the signature which was matched.

Motivation:

This is to help provide clues as to what formats users might have even if DROID cannot identify it precisely. The use case being that users might have a variant of a format that we have in PRONOM but which we haven't seen to allow us to develop a signature.

Solutions:

Notes:

This might be related, and a potential solution for requirement0017 but I am just leaving this separate at the moment while we look for thoughts from the wiki community.
- perhaps we could revisit the hierarchical model, and/or the low confidence signatures model for this requirement? If we see multiple sigs, can we go up a level and just report a family type? - jaygattuso
- DROID is already probabilistic in its identification, in that it looks for (usually quite short) sequences embedded in files in some relation to each other. So DROID never *precisely* identifies anything in the first place - it's format characterization, not validation. Is this a call for having less accurate versions of the signatures, rather than a change to the software? - MattPalmer Jan 24, 2012
- Agreed - my comment belongs in the in PRONOM datamodel space, not the D7 requirements. - JayGattuso Jan 25, 2012

Source:

- rspencer_tna Jan 20, 2012

Declare Interest:

NLNZ

NLA - JonathanMcCabe1 **Jan 26, 2012**

DROID scans DB/EDRMS BLOBS

Description:

It is possible to connect DROID to a DB and use it to scan BLOBS of data and report on their contents.

Motivation:

To increase the range of file stores that DROID is capable of scanning. To allow institutions to easily run DROID on their EDRMS system and have an understanding of the format content which is on there.

Solutions:

Notes:

This was a desired feature during DROID 6 development, but it was shelved as a lot of work - which is not to say it wouldn't be useful. The main difficulties were providing a way to configure connections to a large variety of different database systems (probably using JDBC), specifying the SQL to run to acquire the blobs, defining a URI which was meaningful for files acquired from those systems and which allowed them to be related back to the location in the DBMS they were profiled from (which also involves specifying possibly different SQL to extract some kind of unique primary key/identifier and storing that too in DROID's profile database There was also the issue of extracting other metadata (e.g. date/time) from the systems where it existed - yet more SQL. Finally, this needed to be pausable / resumable even after opening an saved profile. In the end, I think we came to the conclusion that it would be much, much easier to create custom scripts to extract this data from a DBMS to file system for any system you really wanted to do this for, then just run DROID over the file system data that was extracted. Another idea which was considered was to create a JCR (Java Content Repository) interface for DROID, which abstracts out a lot of the details of the underlying DBMS systems and makes content repositories appear much like file systems. Of course, you have to have a JCR connector to your specific content repositories, but these exist for well known content repositories already. - MattPalmer Apr 14, 2012

Source:

- rspencer_tna Jan 20, 2012

Declare Interest:

- **rspencer_tna Jan 20, 2012**

NLNZ

The installation directory for DROID 7 is user configurable

Description:

Users can choose where they want to store installation files for DROID (including signature files, other config files etc.)

Motivation:

For some users this will help get around write access issues for others it is simply an opportunity to override the defaults should they choose to do so.

Solutions:

- Important to see this change within a GUI / installer rather than simply a config file edit

Notes:

Source:

- rspencer_tna Jan 20, 2012

Declare Interest:

- **rspencer_tna Jan 20, 2012**

DROID 7 scans PST files and reports on the format content of those files

Description:

DROID scans PST files and reports on the format content of those files.

Motivation:

We have seen PST files within accessions before and it is difficult to know the precise contents of attachments.

- Can we extend this to eml files? (same problem) - jaygattuso

Solutions:

There is a reasonable Java library for reading PST files (both 32 and 64 bit) here: java-libpst. The DROID architecture makes this fairly straightforward, as all the archive handlers are modular and are very similar to each other. Eml files can also be read in the same way, but using the standard JavaMail package. However, (correct me if I'm wrong), I believe eml files are not entirely standardised... - MattPalmer Jan 24, 2012

- As the number and types of different archive handlers rises (currently only zip, tar and gzip), we will probably want to be able to configure whether each runs or not individually. Right now, you must either process all provided archive files, or none of them - MattPalmer Jan 27, 2012

Notes:

Source:

- rspencer_tna Jan 20, 2012

Declare Interest:

- rspencer_tna **Jan 20, 2012**

NLNZ
NLA - JonathanMcCabe1 **Jan 26, 2012**
BL

ETH
ExLibris

DROID 7 report breakdown reports in configurable units (bytes/kb/mb etc.)

Description:

DROID reports can be configured to report in bytes / kilobytes / megabytes as required by the user to aid in their interpretation of results and reports.

Motivation:

This is to make interpretation of reports directly from DROID easier and to remove one step of analysis for users.

Solutions:

This can probably be achieved already without modifying the software at all. DROID reports are actually XSLT transforms of the DROID report XML format. You can directly modify these XSLT transforms (or create new ones), which could re-calculate the raw sizes in Kb, Mb or Gb. Have a look in the .droid folder in the report_definitions sub-folder. Having said that, providing direct support for selecting the units in the report generation directly would probably be more generally useful, in that it could apply to any report without fiddling with XSLT files! - MattPalmer Jan 24, 2012

Notes:

Source:

- rspencer_tna Jan 20, 2012

Declare Interest:

- rspencer_tna **Jan 20, 2012**

NLA - JonathanMcCabe1 **Jan 26, 2012**

Requirement 0025

Description: Drag and drop

Motivation:
To make the desktop version easier to use

Solutions:

Notes:
-interested in being able to drag folder/set of files onto a new tab, rather than browsing for the folder location
Source:

Declare Interest:
NLNZ

DROID displays the signature file used per each profile tab

Description:

Clear indication of the signature file in use in the GUI

Motivation:

To ensure that users have a visual reference for the signature file currently being used by DROID

Solutions:

Notes:

- Remember each profile tab could be using a different signature file, so it should be clearly bound to the current visible tab somehow. It would be quite handy to be able to see detailed profile configuration and metadata too (probably a property window for that though). - MattPalmer Jan 24, 2012

- Should this apply to reports more clearly as well? - rspencer_tna Jan 25, 2012

- we don't use it, but that would probably make sense... - JayGattuso Jan 25, 2012

Source:

NLNZ

Declare Interest:

NLNZ
NLA - JonathanMcCabe1 **Jan 26, 2012**
ETH

DROID supports the testing of local signatures

Description:

Develop a method for allow the simple testing of local signatures

Motivation:

To support 3rd party signature creation

Solutions:

Notes:

-**WARNING -SCOPE CREEP!** - it would be great to have a method of allowing local signatures to be developed, and to be simply added to the current signature file, without having to extend the existing signature XML. The output from the signature devo tool could be ingested into the local DROID implementation to support rapid sig testing.

- I think there are several aspects to this:
1) A signature developer tool, allowing the creating and testing of new signatures.
2) A signature manager tool, allowing the addition or removal of selected signatures.
3) Submission of generally useful signatures to the National Archives for wider dissemination.
4) The ability to define and use local PUIDs for signatures which are local and are not generally useful. This may mean some kind of namespace for PUIDs. - MattPalmer Jan 24, 2012

-Namespaces for PUIDs! Yes please! - I wrote half a paper on this a while ago that I think I sent to Ross.... The primary argument for being the notion of a format that sits below a format - a variant if you will. There is sometimes some subtle but accessible differences in content from different creation apps, and this would provide a method of locally reflecting that, whilst remaining inside the more generic PRONOM scheme (e.g. fmt/44:NLNZ1 - strip the NLNZ1 and I am still 'compliant' with PRONOM schema. This could be done through local tagging I guess, but it it extends the characterization aspects of DROID, especially where DROID is an embedded tool inside a larger product/workflow. - JayGattuso Jan 25, 2012
- This would be great. - anjacks0n

Source:

NLNZ

Declare Interest:

NLNZ
BL

Signature files can be managed to be more modular and support local variations

Description:

Rethink how signature files are managed, supporting more modular / local variations to be deployed

Motivation:

Flexibility of signature usage

Solutions:

Notes:

- this would allow local users to refine their supported signatures, e.g. I might choose to implement all the tif PUIDS or only the parent fmt/353 puid. This would require some careful management, and deviate from the current master signature list model, but would offer some great local flexibility.This should not result in an overbearing end user setup, but a clear way of dealing with families of content types. It would also speed up processing for deployments that only see a few format types, reducing the workload of the pattern searching process

Another case for PUID namespaces, so local variations can be accomodated alongside signatures from TNA (or other sources of signatures of local or thematic interest). - MattPalmer Apr 14, 2012

Source:

NLNZ

Declare Interest:

NLNZ

DROID supports the local ignoring or overwriting of specific signatures

Description:

Support the local 'ignoring' or overwriting of specific signatures

Motivation:

To ensure that end users have control over the specific signatures that are used where their view of the signature difffers from the PRONOM
Solutions:

Notes:

-Example, fmt/44 now has a small variable offset for the EOF pattern. By using the above method, local users could either extend or remove this offset, depending on thier own institutional view of the expected pattern for the EOF location.

Source:

NLNZ

Declare Interest:

NLNZ

DROID remembers folder last scanned in folder structure

Description:

Setting where DROID looks for content based on last use

Motivation:

To stop the waiting for 'my documents' to be delivered to the file chooser when 'my docs' is not local and slow...

Solutions:

Notes:

- I mainly notice this for uploading sig files, not the content chooser.- JayGattuso Jan 25, 2012
- Edited the title. I hope this reflects accurately what was being asked :) - rspencer_tna Jan 26, 2012

Source:

Declare Interest:

NLNZ

DROID Command line is easier to use

Description:
Improve ease of using the DROID command-line.

Motivation:
At present, running DROID from the command line forces you to do two runs. You have to first create a profile. Then you have to load the profile back in in order to export or report on it. It would be much nicer if you could specify an export or report directly along with specifying what to profile. It would also be nice if command line DROID could simply output its results to the console as they appear, instead of requiring the database (see also requirement0016)

Solutions:

Notes:
(I confess, the DROID command line was not given as much love as it needed during DROID 5 and 6 development. But it's a much simpler project than the GUI, and it would not be difficult to significantly improve it, with very little effort - MattPalmer Jan 27, 2012).
- Would be great to have a smaller, neater command-line client. Ideally, a the signature update would be optional (for speed), and passing a different signature set would be a command-line option. - anjacks0n

Source:
- MattPalmer Jan 27, 2012

Declare Interest:

BL
NLNZ
NANETH

DROID identifies the creating application / environment of files

Description: DROID should be able to identify the application and or environment which was used to create a file.

Motivation: Knowing the creating application/environment is valuable for at least two reasons:
1. it provides a strong indication as to the best environment to render the object in which is useful for 2 additional reasons:
a) the original rendering environment can be used as a control to validate migration actions against.
b) Information about the original rendering environment is needed to enable emulation as a preservation strategy.
2. The creating application can change how a file is structured, knowing which environment was used to create a file could potentially be as valuable as knowing which formatting standard the (primary) application creator was intending to adhere to when writing out the file. Knowing which environment was used to create a file could help to, for example, make the ID process more efficient by enabling DROID to skip other parts of the ID process based on that knowledge.

Solutions: Some types of files have a specific location that can be used by application creators to note which application and version were used to create the file (e.g. tag 305 in the TIFF 6.0 standard). Others may be able to have this information inferred through a pattern matching process the same as that used to identify the file formatting standard that the application writers were intending to adhere to.

Notes: There is anecdotal evidence of the potential for a lot of false positives due to application creators not changing embedded metadata fields stating the creating application when e.g. re-purposing opensource code or upgrading versions and changing other aspects of the way files are written out.

This could effectively result in the need for sub-formats or format variants or new formats that are defined by the combination of the creating environment and the formatting standard used. - EuanCochrane

I think this is something which could be handled using custom signatures tailored to look for these distinctive patterns. As such, it's probably not a change to DROID itself - although it does strengthen the case for a generally usable signature management application and also for the ability to define local PUIDs (and PUID namespaces), so people can create signature sets tailored to look for whichever characteristics they are interested in (and also to share them with others). - MattPalmer Apr 14, 2012

Source: Discussion about this report

Declare Interest: **Archives NZ**

DROID is able to load a list of files to identify from a file

Description:
DROID should be able to load a list of files which it has to identify from a list (one line per file).

Motivation:
At the moment the only way to pass a list of files to identify is by passing filenames with the "-a" argument to have them added to a profile. This causes problems if the list is larger than the maximum allowed length of a single command.

The reason for this is that sometimes we want to compile a list of files to identify.
An example use case would be to search a drive for all "*.doc" files and have them identified instead of identifying all files on that drive using recursion.

Solutions:
Workaround for CLI: pass a limited number of filenames to have them added to the profile and repeat this.
Workaround for GUI: select files one at a time by browsing.
Both workarounds are very tedious.

Possible solution for CLI: add argument to DROID to load a list of files
Possible solution for GUI: add button to select and load a list of files

Notes:
On computers running Microsoft Windows XP or later, the maximum length of the string that you can use at the command prompt is 8191 characters. On computers running Microsoft Windows 2000 or Windows NT 4.0, the maximum length of the string that you can use at the command prompt is 2047 characters. (http://support.microsoft.com/kb/830473)

Source:
- techmaurice_naneth Feb 23, 2012

Declare Interest:
NANETH

DROID Can generate a range of checksums for users

Description:

DROID offers alternative checksum types for users. MD5, SHA-1, SHA-256 etc.

Motivation:

We can potential use this mechanism in ingest, we might want to future-proof it for any potential requirements for more secure, different hashes.

Question: do you mean being able to provide more than one checksum for each file using a range of algorithms, or to be able to select different hash algorithms to use (but only one checksum is calculated for each file)? - MattPalmer Apr 14, 2012

Solutions:

Notes:

We are already seeing records that aren't using MD5.

Allowing the use of different hash algorithms is a good idea. It's trivial to implement different hash algorithms in the current architecture - the burden of work falls on providing user interfaces to configure it and persisting the setting in the profile. However, I question what you mean by "more secure" in the context of ingest. The (original) goal of providing MD5 hashes was to easily detect corruption and/or accidental substitution in a file being ingested. For that purpose, MD5 is ideal, as it is extremely fast. It is very insecure if you are trying to detect malicious substitution of a file, as it is relatively easy to intentionally create different files which have the same MD5 hash (but this is extremely unlikely to happen by accident). These attacks are called pre-image attacks. There are two kinds:

1. first pre-image attack: given a hash of a file, find another file which produces the same hash.
2. second pre-image attack: given a file, find a different file which produces the same hash.

Although there are still no really practical attacks of either sort against SHA-1, it is not recommended to use if you have the choice, as weaknesses have been discovered - particularly if you are relying on these hashes post-ingest for long periods of time (as might be the case in an archive). For the time being, SHA-256 would be a better choice if you concerned about either of these two possibilities. In the near future, the SHA-3 family of hashes will be come available, which should be even more secure, although as ever, it's usually best to wait a while before leaping on the latest and greatest cryptographic magic, just in case. - MattPalmer Apr 14, 2012

Source:

- rspencer_tna Apr 13, 2012

Declare Interest:

- rspencer_tna Apr 13, 2012

Folder metadata via Active Directory hooks into / from DROID

Description:

Use / Explore the ability to hood DROID into active directory to retrieve more metadata by hooking into Active Directory.

Motivation:

Retrieve more metadata about files.

Solutions:

Notes:

This proposal is a bit unclear on the kind of additional metadata which would be useful to retrieve. What *specific* information on files or folders does Active Directory have which makes it worthwhile providing an integration to a single directory technology? - MattPalmer Jul 25, 2012

Would have to explore what might be provided by active directory, however the suggestion from users is it could provide a reasonable idea about users and folders/files and therefore allow decisions to be made on the sensitivity/security of data. However, it is noted, this might be beyond what users require DROID for. Really it's a suggestion to investigate it. Although any clearer idea from experts with this knowledge on this page is appreciated. - rspencer_tna Jul 25, 2012

Information about users and permissions can be obtained from the new file access package in java 7. It can obtain information about access control lists for NTFS file systems - which includes the users, and also POSIX file system users and permissions. This does not depend on a directory technology, but it would only work on JRE 7 or above. In fact, as far as I know, Active Directory doesn't typically store information about files and folders at all, although I'm prepared to be wrong about that as I'm not an AD expert :) - MattPalmer Jul 25, 2012

Source:

- rspencer_tna Jul 25, 2012

Declare Interest:

- rspencer_tna Jul 25, 2012

DROID better supports the development of new signatures

Description:
The workflow within DROID for adding and testing custom signatures has a lot of potential for improvement

Motivation:
Currently it is quite a hazzle to develop and test new custom signatures within DROID. In order to use a custom signature file,

1. the version number specified by the Version attribute in the FFSignatureFile element of the signature file has to be set/updated (see issue 13 in the DROID issue list at github)
2. the file has to be added as a new signature
3. the new signature has to be selected as the current signature (issue 12)
4. a new profile as to be opened
5. the files to check have to be added
6. and the identification has to be started.

And for every modification of the signature file this entire process has to be again.

Solutions:
A signature development tool as proposed in requirement0027: DROID supports the testing of local signatures should certainly help, but there are a few rather simple modifications in DROID which would make the entire workflow a lot easier.

1. Fix issue 12 and issue 13 in the DROID issue list at github
2. When adding a new signature, the user can choose (e.g. via a checkbox) whether the signature file should be saved within DROID (just like it does now) or if the signature file should just be linked to. This way the user can continue to edit the signature file at the current location (e.g. the users documents folder) and every time an identification is run with this signature, the current version of the linked file is used.
    1. Another way of achieving this would be to allow DROID to have a list of signature file locations it will use. At present it just defaults to the signature files stored under the .droid folder. If you could configure this location, or allow multiple signature file locations to be specified, then you could have built-in and editable signatures available to DROID. In fact, many aspects of where droid stores and consumes information is already configurable in this way. This might be easier to do and more consistent and generic than allowing specific signature files to be "linked". - MattPalmer Nov 9, 2012
3. The signature file used for an identification can be changed after a new profile is created. Also it should be easier to select with what signature file the current profile should be identified. This could be done by making the Start-button into a split button (just like the Run-button in Eclipse). If the user just presses the button normally the current default signature file is also used for the new identification (just like it is right now). But if the user clicks the button extension, a list of all signatures is displayed (and the current default signature is marked) and the user can select any of

these signatures to run the identification with (thereby making it the new default signature). See the simple mockup for how such a button could look like.

1. Nice idea to allow changing which signature is used in a profile after it is created. In fact, most profile properties would benefit from being editable after profile creation. It's really common to create a new profile, and only then remember you wanted to turn off going inside zip files, or whatever. The only small issue with allowing such changes is if the change is allowed after a profile has been started. If you allow changes after this point, then the profile metadata ceases to make sense, and you've have to have some kind of profile history. There would be a lot of (relatively minor) changes to other parts of the software if changes post-starting were allowed. - MattPalmer Nov 9, 2012
2. The GUI suggested above makes the rather large assumption that there are only binary signature files, and it places front-and-center an option which I believe is only of interest to relatively few users of DROID (I could be wrong about that...). There are also container signature files, and potentially other sorts (e.g. if some sort of text heuristic signatures were added in future). Personally, I would rather see a new profile properties button which allowed the editing of all profile properties, at least up to the point the profile is started. - MattPalmer Nov 9, 2012

4. After an identification is done, there should be the option to reidentify the current profile. This can simply be done by reactivating the Start-Button after the identification is finished. This way the user can either run a different signature file on the same set or run the same file which might have been changed in the mean time (if it is a linked signature file as described in step 2)
    - I guess it would also be helpful (though less important) to have the rerun entry in the context menu when right clicking on a file or folder so that individual files and folders can be reidentified without having to run it on all the files again.
    - This is a bit of a can of worms. It was considered but ruled out in the original DROID 5 and 6 development. It's such a large topic, that it really deserves its own requirement - MattPalmer Nov 9, 2012
        - First a small point: if the start button is re-enabled when the profile is finished, it looks like the profile hasn't finished. I would tend to try to avoid overloading the start button too much - maybe a menu option or something (it would be a lesser used option, after all), to "re-profile". - MattPalmer Nov 9, 2012
        - Second: if files have changed since the original profile, what do you do with them? A file which was in the original profile but is no longer there? A new file? A change in identification because you changed the signature file used? How do you flag change in the GUI, reports and exports? Building that level of profile history into DROID would be a massive change to its GUI, internal data model, reporting and querying capability. - MattPalmer Nov 9, 2012
        - Is the goal to be able to compare the old and new profile data? If so, it may be better just to start a new profile, then compare the exports of both profiles. This is much simpler to achieve. If the goal is not to compare, but just to "wipe all contents and start again", then you're really just creating a completely new profile that happens to have the same top-level file/folder definitions. So again, re-profiling

really means here: create a new profile using the high level definitions of the original one as a starting point. I think this is quite a good idea, and much, much, easier to achieve. - MattPalmer Nov 9, 2012

All of these measures will make it a lot easier to work on new signatures, which might encourage people to create signatures for the files they are working with an contributing them back to TNA (see point 3 proposed
in requirement0027: DROID supports the testing of local signatures)

Notes:

Source:
- DavidFichtmueller Nov 9, 2012

Declare Interest:
- DavidFichtmueller Nov 9, 2012

Plug-ins can be added to DROID

Description:
It should be possible for other developers do create plug-ins for DROID to add special functionality which is not part of the core focus of the general DROID development.

Motivation:
DROID is used by very different actors in a lot of different contexts with a lot of different needs and approaches. There are several specific features some institutions or individuals might want to have in DROID but that are not and will not be implemented, because they are not part of the core focus of DROID and also might have negative effects on people using DROID but not these specific features (primarily slowing DROID down). A plug-in architecture might be a good solution for these cases. Institutions (or individuals) could create/use DROID plug-ins for their specific needs whereas the users who don't need this functionality just don't install/activate the plug-in and have no negative effects because of this.

Solutions:
Adding a plug-in architecture to an existing software can be quite challenging. To keep it simple, I would propose only one specific kind of plug-in which extends the identification capabilities of DROID.

A plug-in can specify if it is interested in some specific file formats (by providing a list of PUIDs) or in all files. After DROID is done with the identification of a particular file, it checks if there are any plug-ins which are interested in this file and if so it passes it to this particular plug-in. When a plug-in is called, it gets handed the current file and the identification result of this file. The plug-in specific code is then run and at the end the plug-in can either

- return the identification unchanged
- add its own identification to the list of identifications
- overwrite the previous identification
- or add an additional information field which will be displayed in a new column in the result table

In the preferences window the user can see the plug-ins that are currently installed, add new plug-ins, change the order in which they are applied or (de)activate specific plug-ins if their functionality is (not) needed for the next identification. Maybe a plug-in could also specify what settings can be set for it, this however will make it a lot more complicated to implement the plug-in functionality. This could be a feature which would be implemented later on.

From a technical point of view a plug-in can be created by implementing a particular Java interface. The compiled class could then be put in a jar-file and placed in a specific plug-in folder in the DROID installation directory. When DROID is started it looks in this folder for classes implementing the plug-in interface and includes them.

Since some of the features that will be implemented might also be interesting to other institutions/users, existing plug-ins could be added to some kind of online directory or market place so they can be downloaded and used by others.

Notes:
I have looked through the list of features proposed in this wiki and identified several features which could very well be implemented as plug-ins:

- 0004: DROID contains mechanisms for format validation
- 0005: DROID identifies attempts at data obfuscation (passwords / encryption / etc.)
- 0012: DROID can identify formats using the Java native regex library (though this would require an additional kind of signature file)
- 0013: DROID can identify text based formats such as source code and scripting languages (though this would require an additional kind of signature file)
- 0034: DROID Can generate a range of checksums for users

Source:
- DavidFichtmueller Nov 9, 2012

Declare Interest:
- DavidFichtmueller Nov 9, 2012

Ability to check against one fmt given from a partner through a local test

Description:
When exchanging a file between a partner (client) and an electronic archive system, we would like to check the fmt given by the partner against one file with Droid to check if the given fmt is correct. To be able to do that, we have to face various situations:
- the fmt given is perfectly designated and so the answer is the same locally
- the fmt given is totally wrong (see next point to identify if it is really wrong)
- the fmt given is ok but relies on an older signature file and therefore our Droid gives us a new fmt instead of the given one. In this case, it could appear today as a totally wrong format identification. But in most cases, in fact, the fmt given is a "sub-fmt", meaning it is an old one that was overriden by a new one (more precise). This appears for instance recently with Word 2007/2010 formats. We would like that in this case Droid gives us back the correct current fmt but with not an error but a warning.

Motivation:
The motivation is based on exchanging files between partners and using Droid to have one more check to validate this exchange.

Solutions:
Currently we change the Droid behaviour to give us back not the only top fmt found, but all fmt found, with at first place the correct current one.
For instance in our current test implementation, what we have done is to change a bit the ResultPrinter module to have the following logic:
1) before calling removeLowerPriorityGits, we backup the results as it is
2) we continue the logic as today
3) when printing the result out, we check if the backup list is empty, and if not we print: fileName,realPuid,backupList, else we print as before fileName,realPuid

Doing this let the caller the possibility to parse the list and check if a given fmt is in the list, in the first position or not, therefore answering the question.
You can have a look at our current test implementation in:
https://github.com/fredericBregier/VitamTools/tree/master/src/main/java/uk/gov/nationalarchives/droid/command
Note however that this tested implementation does a little more than needed and moreover its implementation might be largely improved as it is only a "proof of concept" implementation.

Notes:

Source:

Declare Interest:
French NA

Add Bzip2ArchiveContentIdentifier

Description:
As there are already ZIP, Tar, GZIP, it is easy to add Bzip2ArchiveContentIdentifier following the exact same implementation than GZIP.

Motivation:
From some partners, in particular those in "recent" Linux, the IT team often use tar.bzip2 as format of container, and not tar.gzip due to higher compression capability.
As in Droid all the necessary dependency is already there (Apache Compress), it is really easy to add this in Droid.

Solutions:
The solution is to create a copy of GzipArchiveContentIdentifier, replacing the necessary element to use Bzip2 instead of Gzip. We've tested it and it works well.
You can have a look at our current test implementation in :
https://github.com/fredericBregier/VitamTools/tree/master/src/main/java/uk/gov/nationalarchives/droid/command/archive

Notes:

Source:

Declare Interest:
French NA

Having the possibility to get the FileFormat from a puid or a filename extension directly without "hacking" Droid internal data structures

Description:
Often we need to have access to the information of the FileFormat, and not only to the PUID, or to be able to get a default PUID from the extension of the filename when no puid was found using the standard way. To be able to do so, we need to have a method that make an abstraction on the way the signatures are stored and however a way to get information back from them.

Motivation:
Currently, when we ask for a format identification, sometime the format is not recognized. We use then the internal methods of FFSignatureFile (getTentativeFormatsForExtension or getFileFormatsForExtension) that gives us a result. For instance, EML files are only detected like this.
A second case is when we have the puid as an answer (given by "filename,puid,..." result), we would like to have more description from the FileFormat (mimetype, other information). Today, to be able to do so, we have to use the internal method from FFSignatureFile (getFileFormat).
Both solutions are working but with 2 limitations:
1) we had to make "public" the getter of the attribute FFSignatureFile from the BinarySignatureIdentifier.
2) we had to access then the internal data structure of Droid (FFSignatureFile) to get those method availables.

This is not I think a good practive. Droid should mask the below implementation (which is done today) but then provide a way to request such methods through global configuration object, such as from a "SIgnatureIdentifier", that could be below a BinarySignatureIdentifier or something else.

Solutions:
Provide methods in the SIgnatureIdentifier global object (or another one if better) that allows to request the content of a specific FileFormat through its puid, and to provide a way to check with a method in the same object the signature through the extension of a specific file (usefull when no identification was provided at all by Droid).

Notes:

Source:

Declare Interest:
French NA

Having the possibility to exclude x-fmt puid from answers

Description:
While x-fmt give us a way to identify however a file format, this looks like as a "not" official format. Therefore we would like to have a way to specify that we would like to exclude such results from Droid when identifying a file format.

Motivation:
The motivation is to stay with "official" mimetypes, while x- format are not yet standardized formats and therefore seem to be not a correct answer from "legal" perspective.

Solutions:
Having a way to exclude through the API those x-fmt from the results through an option.
Note however that a second possibility (what we have done for the moment) is to get all "puid" related to one file, not only the highest priority one, and then filtering the output by removing the x-fmt.

Notes:

Source:

Declare Interest:
French NA

Mandatory match of signature and extension

Description:
Currently if a file matches a signature but not the extension that is assigned to the signature, DROID will match the signature and show a warning that the file has the wrong extension. Whereas if the file extension matches but not the signature (and no other signature), DROID will display the result and will shown "Extension" as the method of identification. There should however be a way to specify that a file type should only be matched if both the signature and the extension apply to the file in question.

Motivation:
There are a lot of file types out there that are really suitable for signature matching because the few parts that all of the files have in common are very common among other file types as well. Just as there are a lot of file types out there that use the same file extensions. However, with the combination of these two factors it would be possible to safely match these file types with a lot less false positives.

Solutions:
The solution would be to add an optional attribute to the XML schema for the element. The attribute could be called "Match" and it would have the two allowed values "loose" and "strict", whereby "loose" would be the default value. So if the attribute is not specified, it would be the same if it was specified with the value "loose" and the file format would be matched if either the signature or the file extension matches, just as it is now.
An example could look like this:

378
cdr

It's an interesting proposal, but I'm not sure you could encode this for specific file formats for all users. I think different communities will have a different view of which formats require this sort of super-strict matching or not. If you are only interested in matching formats with no extension mismatches, you can always just filter the results to exclude those with extension mismatches. Or have I missed something here? - MattPalmer Dec 20, 2012

Notes:
The suggested names are just that, only suggestions. Fell free to change them if you find other terms more suitable.
This approach would also be backward compatible to all the old signature files.

Source:
- DavidFichtmueller Dec 20, 2012

Declare Interest:
- DavidFichtmueller Dec 20, 2012

DROID flags files that are part of analysed container files

Description:

Flag where a file listed is in an a container that DROID has scanned the contents of.

Motivation:

To provide better filtering mechanisms, i.e. currently a DROID report is more difficult to trim down if container contents have been analysed to generate file counts of 'top-level' items.

Solutions:

- An additional column in the DROID report could work.

Notes:

N/A

Source:

Archives New Zealand - - rspencer_tna Apr 29, 2014

Declare Interest:

Archives New Zealand - rspencer_tna Apr 29, 2014

Requirement 0044

Description:

Motivation:

Solutions:

Notes:

Source:

Declare Interest:

Requirement 0045

Description:

Motivation:

Solutions:

Notes:

Source:

Declare Interest:

Requirement 0046

Description:

Motivation:

Solutions:

Notes:

Source:

Declare Interest:

Requirement 0048

Description:

Motivation:

Solutions:

Notes:

Source:

Declare Interest:

Requirement 0049

Description:

Motivation:

Solutions:

Notes:

Source:

Declare Interest:

Requirement 0050

Description:

Motivation:

Solutions:

Notes:

Source:

Declare Interest:

Requirements Catalogue User Guide

Each requirement in the catalogue is divided into six sections. Each section can be used as follows:

Description:

A brief description of the requirement. This should normally be expressed in the present tense. An example might be 'DROID identifies the contents of WARC files'. You may wish to be as verbose as you need however. The description may be amended or edited to make it consistent with the rest of the catalogue, The National Archives will make it clear when this is done.

Motivation:

The motivation field gives users the opportunity to expand on why they have a particular requirement. This acts like a use case which can help us to refine the requirement further and understand it in its full context. Users from multiple organisations may add multiple motivations.

Solutions:

If there is a known solution to a problem you can link to it here. This might involve software libraries, complete software packages or just preliminary research into a topic. There is no limit into what you might post here as any solutions may be immediately useful to another user.

Notes:

Additional notes, or suggestions that don't quite fit anywhere else can be put here.

Source:

It helps if the user creating the requirement declares they created it here. This can be done by typing four tilde (~) characters, This shows up when saved like: - rspencer_tna Nov 17, 2011

Declare Interest:

Additional users with a particular interest in seeing a requirement reach DROID can declare themselves here. The popularity of a requirement will help The National Archives determine how to approach focussing the requirements catalogue before development. To add your username follow the step above and type four tilde (~) characters.

To add a requirement simply do the following:

1. find an empty placeholder page.
2. Edit the main title in the WYSIWYG editor
3. Fill in the additional fields and click save.

Users may also choose to update the placeholder title on the index page. You can do this by editing the

page, highlighting the link and clicking 'change'. The National Archives will endeavour to ensure there is always room for additional requirements.

For additional help with editing pages, users may choose to look at the wikispaces guide on this subject here: http://help.wikispaces.com/Wikitext