Table of Contents

Chapter 1. About

This document contains more examples of resolving compact identifier-based DRS URIs than we could fit in the DRS specification or appendix. It's provided here for your reference as a supplement to the specification.

Chapter 2. Background on Compact Identifier-Based URIs

Compact identifiers refer to locally-unique persistent identifiers that have been namespaced to provide global uniqueness. See "Uniform resolution of compact identifiers for biomedical data" for an excellent introduction to this topic. By using compact identifiers in DRS URIs, along with a resolver registry (identifiers.org/n2t.net), systems can identify the current resolver when they need to translate a DRS URI into a fetchable URL. This allows a project to issue compact identifiers in DRS URIs and not be concerned if the project name or DRS hostname changes in the future, the current resolver can always be found through the identifiers.org/n2t.net registries. Together the identifiers.org/n2t.net systems support the resolver lookup for over 700 compact identifiers formats used in the research community, making it possible for a DRS server to use any of these as DRS IDs (or to register a new compact identifier type and resolver service of their own).

We use a DRS URI scheme rather than Compact URIs (CURIEs) directly since we feel that systems consuming DRS objects will be able to better differentiate a DRS URI. CURIEs are widely used in the research community and we feel the fact that they can point to a wide variety of entities (HTML documents, PDFs, identities in data models, etc) makes it more difficult for systems to unambiguously identify entities as DRS objects.

Still, to make compact identifiers work in DRS URIs we leverage the CURIE format used by identifiers.org/n2t.net. Compact identifiers have the form:

```
prefix:accession
```

The prefix can be divided into a `provider_code` (optional) and `namespace`. The `accession` here is an Ark, DOI, Data GUID, or another issuers's local ID for the object being pointed to:

```
[provider_code/]namespace:accession
```

Both the `provider_code` and `namespace` disallow spaces or punctuation, only lowercase alphanumerical characters, underscores and dots are allowed.

Examples include (from n2t.net):

```
PDB:2gc4
Taxon:9606
DOI:10.5281/ZENODO.1289856
ark:/47881/m6g15z54
IGSN:SSH000SUA
```

TIP	DRS URIs using compact identifiers with resolvers registered in identifiers.org/n2t.net can be distinguished from the hostname-based DRS URIs below based on the required ":" which is not allowed in hostname-based URI.

See the documentation on n2t.net and identifiers.org for much more information on the compact identifiers used there and details about the resolution process.

Chapter 3. Example DRS Client Compact Identifier-Based URI Resolution Process - Existing Compact Identifier Provider

A DRS client identifies the a DRS URI compact identifier components using the first occurance of "/" (optional) and ":" characters. These are not allowed inside the provider_code (optional) or the namespace. The ":" character is not allowed in a Hostname-based DRS URI, providing a convenient mechanism to differentiate them. Once the provider_code (optional) and namespace are extracted from a DRS compact identifier-based URI, a client can use services on identifiers.org to identify available resolvers.

Let's look at a specific example DRS compact identifier-based URI that uses DOIs, a popular compact identifier, and walk through the process that a client would use to resolve it. Keep in mind, the resolution process is the same from the client perspective if a given DRS server is using an existing compact identifier type (DOIs, ARKs, Data GUIDs) or creating their own compact identifier type for their DRS server and registering it on identifiers.org/n2t.net.

Starting with the DRS URI:

```
drs://doi:10.5072/FK2805660V
```

with a namespace of "doi", the following GET request will return information about the namespace:

```
GET
https://registry.api.identifiers.org/restApi/namespaces/search/findByPrefix?prefix=doi
```

This information then points to resolvers for the "doi" namespace. This "doi" namespace was assigned a namespace ID of 75 by identifiers.org. This "id" has nothing to do with compact identifier accessions (which are used in the URL pattern as `{$id}` below) or DRS IDs. This namespace ID (75 below) is purely an identifiers.org internal ID for use with their APIs:

```
GET
https://registry.api.identifiers.org/restApi/resources/search/findAllByNamespaceId?id=
75
```

This returns enough information to, ultimately, identify one or more resolvers and each have a URL pattern that, for DRS-supporting systems, provides a URL template for making a successful DRS GET request. For example, the DOI urlPattern is:

```
urlPattern: "https://doi.org/{$id}"
```

And the `{$id}` here refers to the accession from the compact identifier (in this example the

accession is `10.5072/FK2805660V`). If applicable, a provide code can be supplied in the above requests to specify a particular mirror if there are multiple resolvers for this namespace. In the case of DOIs, you only get a single resolver.

Given this information you now know you can make a GET on the URL:

```
GET https://doi.org/10.5072/FK2805660V
```

*The URL above is valid for a DOI object but it is not actually a DRS server! Instead, it redirects to a DRS server through a series of HTTPS redirects. This is likely to be common when working with existing compact identifiers like DOIs or ARKs. Regardless, the redirect should eventually lead to a DRS URL that percent-encodes the accession as a DRS ID in a DRS object API call. For a **hypothetical** example, here's what a redirect to a DRS API URL might ultimately look. A client doesn't have to do anything other than follow the HTTPS redirects. The link between the DOI resolver on doi.org and the DRS server URL below is the result of the DRS server registering their data objects with a DOI issuer.*

```
GET https://drs.example.org/ga4gh/drs/v1/objects/10.5072%2FFK2805660V
```

IDs in DRS hostname-based URIs/URLs are always percent-encoded to eliminate ambiguity even though the DRS compact identifier-based URIs and the identifier.orgs API do not percent-encode accessions. This was done in order to 1) follow the CURIE conventions of identifiers.org/n2t.net for compact identifier-based DRS URIs and 2) to aid in readability for users who understand they are working with compact identifiers. **The general rule of thumb, when using a compact identifier accession as a DRS ID in a DRS API call, make sure to percent-encode it. An easy way for a DRS client to handle this is to get the initial DRS object JSON response from whatever redirects the compact identifier resolves to, then look for the `self_uri` in the JSON, which will give you the correctly percent-encoded DRS ID for subsequent DRS API calls such as the `access` method.**

Chapter 4. Example DRS Client Compact Identifier-Based URI Resolution Process - Registering a new Compact Identifier for Your DRS Server

See the documentation on n2t.net and identifiers.org for adding your own compact identifier type and registering your DRS server as a resolver. We document this in more detail in the main specification document.

Now the question is how does a client resolve your newly registered compact identifier for your DRS server? *It turns out, whether specific to a DRS implementation or using existing compact identifiers like ARKs or DOIs, the DRS client resolution process for compact identifier-based URIs is exactly the same.* We briefly run through process below for a new compact identifier as an example but, again, a client will not need to do anything different from the resolution process documented in "DRS Client Compact Identifier-Based URI Resolution Process - Existing Compact Identifier Provider".

Now we can issue DRS URI for our data objects like:

```
drs://mydrsprefix:12345
```

This is a little simpler than working with DOIs or other existing compact identifier issuers out there since we can create our own IDs and not have to allocate them through a third-party service (see "Issuing Existing Compact Identifiers for Use with Your DRS Server" below).

With a namespace of "mydrsprefix", the following GET request will return information about the namespace:

```
GET
https://registry.api.identifiers.org/restApi/namespaces/search/findByPrefix?prefix=myd
rsprefix
```

Of course, this is a hypothetical example so the actual API call won't work but you can see the GET request is identical to "DRS Client Compact Identifier-Based URI Resolution Process - Existing Compact Identifier Provider".

This information then points to resolvers for the "mydrsprefix" namespace. Hypothetically, this "mydrsprefix" namespace was assigned a namespace ID of 1829 by identifiers.org. This "id" has nothing to do with compact identifier accessions (which are used in the URL pattern as `{$id}` below) or DRS IDs. This namespace ID (1829 below) is purely an identifiers.org internal ID for use with their APIs:

```
GET
https://registry.api.identifiers.org/restApi/resources/search/findAllByNamespaceId?id=
1829
```

Like the previous GET request this URL won't work but you can see the GET request is identical to "DRS Client Compact Identifier-Based URI Resolution Process - Existing Compact Identifier Provider".

This returns enough information to, ultimately, identify one or more resolvers and each have a URL pattern that, for DRS-supporting systems, provides a URL template for making a successful DRS GET request. For example, the "mydrsprefix" urlPattern is:

```
urlPattern: "https://mydrs.server.org/ga4gh/drs/v1/objects/{$id}"
```

And the `{$id}` here refers to the accession from the compact identifier (in this example the accession is `12345`). If applicable, a provide code can be supplied in the above requests to specify a particular mirror if there are multiple resolvers for this namespace.

Given this information you now know you can make a GET on the URL:

```
GET https://mydrs.server.org/ga4gh/drs/v1/objects/12345
```

So, compared to using a third party service like DOIs and ARKs, this would be a direct pointer to a DRS server. However, just as with "DRS Client Compact Identifier-Based URI Resolution Process - Existing Compact Identifier Provider", the client should always be prepared to follow HTTPS redirects.

To summarize, a client resolving a custom compact identifier registered for a single DRS server is actually the same as resolving using a third-party compact identifier service like ARKs or DOIs with a DRS server, just make sure to follow redirects in all cases.

Note: Issuing Existing Compact Identifiers for Use with Your DRS Server

See the documentation on n2t.net and identifiers.org for information about all the compact identifiers that are supported. You can choose to use an existing compact identifier provider for your DRS server, as we did in the example above using DOIs ("DRS Client Compact Identifier-Based URI Resolution Process - Existing Compact Identifier Provider"). Just keep in mind, each provider will have their own approach for generating compact identifiers and associating them with a DRS data object URL. Some compact identifier providers, like DOIs, provide a method whereby you can register in their network and get your own prefix, allowing you to mint your own accessions. Other services, like the University of California's EZID service, provide accounts and a mechanism to mint accessions centrally for each of your data objects. For experimentation we recommend you take a look at the EZID website that allows you to create DOIs and ARKs and associate them with your data object URLs on your DRS server for testing purposes.