Authority Control of Metadata Values

This page documents the new *choice management* and *authority control of Item ("DC") metadata values* features in DSpace 1.6. These are actually two separate features: the *choice management* mechanism is completely independent of authority control and can be configured and deployed independently of it.

If you are wondering what *authority control* is and why it is important please see Dorothea Salo's comprehensive paper, *Name authority control in institutional repositories*.

Note that *this is a general mechanism* for authority control of *any metadata field*: Although author (and other personal) names are the most common (and most urgently needed) example, the prototype can be applied to *any* of the DC fields. It requires some minor, backward-compatible changes to the DSpace 1.5 data model and API. A prototype implementation is available for DSpace 1.6.

Introduction and Motivation

Definitions

- Choice Management - This is a mechanism that generates a list of *choices* for a value to be entered in a given metadata field. Depending on your implementation, the exact choice list might be determined by a *proposed value* or *query*, or it could be a fixed list that is the same for every query. It may also be *closed* (limited to choices produced internally) or *open*, allowing the user-supplied query to be included as a choice.
- Authority Control - This works in *addition* to choice management to supply an *authority key* along with the chosen value, which is also assigned to the Item's metadata field entry. Any authority-controlled field is also inherently choice-controlled.

About Authority Control

The advantages we seek from an authority controlled metadata field are:

1. **There is a simple and positive way to test whether two values are identical**, by comparing *authority keys*.
   - Comparing plain text values can give false positive results e.g. when two different people have a name that is written the same.
   - It can also give false negative results when the same name is written different ways, e.g. "J. Smith" vs. "John Smith".
2. **Help in entering correct metadata values.** The submission and admin UIs may call on the authority to check a proposed value and list possible matches to help the user select one.
3. **Improved interoperability.** By sharing a name authority with another application, your DSpace can interoperate more cleanly with other applications.
   - For example, a DSpace institutional repository sharing a naming authority with the campus social network would let the social network construct a list of all DSpace Items matching the shared author identifier, rather than by error-prone name matching.
   - When the name authority is shared with a campus directory, DSpace can look up the email address of an author to send automatic email about works of theirs submitted by a third party. That author does *not* have to be an EPerson.
4. Authority keys are normally *invisible* in the public web UIs. They are only seen by administrators editing metadata. The value of an authority key is not expected to be meaningful to an end-user or site visitor.

Authority control is *different* from the controlled vocabulary of keywords already implemented in the submission UI:

1. **Authorities are external to DSpace.** The source of authority control is typically an external database or network resource.
   - Plug-in architecture makes it easy to integrate new authorities without modifying any core code.
2. This authority proposal impacts **all phases of metadata management**.
   - The keyword vocabularies are only for the submission UI.
   - Authority control is asserted *everywhere* metadata values are changed, including unattended/batch submission, LNI and SWORD package submission, and the administrative UI.

Some Terminology

- Authority - An *authority* is a source of fixed values for a given domain, each unique value identified by a *key*.
  For example, the OCLC LC Name Authority Service.
- Authority Record - The information associated with one of the values in an authority; may include alternate spellings and equivalent forms of the value, etc.
- Authority Key - An opaque, hopefully persistent, identifier corresponding to exactly one record in the authority.

Choice Management: Design Principles and Behavior

Choice management may be applied to any metadata field in the DSpace configuration properties. This configuration is effective in all community and collection contexts.

Source of Choices

You configure a metadata field for choice management by selecting a *choice authority* plugin for it. This plugin serves as a *source* of choices. Whenever the user is entering a metadata value for that field, e.g. in the interactive submission UI or when editing the Item's metadata, the UI consults that *choice authority* plugin to get a list of available choices to present to the user. This list *may or may not* be affected by the current (or proposed) value of the field.

Presentation Style

You may configure a *presentation style* for the metadata field that governs how the UI displays choices and interacts with the user to pick one.

The available values are:

- lookup - User enters a proposed value and clicks a button to "look up" choices based on that value, and present a pop-up window that lets her navigate through choices.
- suggest - As the user types in a text-input field, a menu of suggested choices is automatically generated. It acts like the Google Suggest feature.
- select - Puts up a drop-down menu (or multi-pick selection box) of choices using the HTML SELECT widget. This style should *only* be used for plugins that have a relatively small, and fixed set of choices. It does *not* support authority values and should not be used for authority-controlled fields.

Authority Control: Design Principles and Behavior

1. **Not a replacement for text metadata value.**Metadata fields still have text values.
   - The text value of a metadata field does *not* have to be derived from the authority, even if authority control is required for that field.
2. **Configured by *field*.** The authority control status of each field is independently configured, but it affects *all* values of that field.
3. **Authority control can be *optional* or *required*.** When optional, metadata values *may* take on values that did not come from the authority.
4. **Authority values are ubiquitous.** Authority values are accessible by crosswalk [1] plugins, in the UI, through OAI-PMH, etc.

   - All of those context can detect whether a value is authority-controlled or not by testing for presence of an authority key.
5. **Text-based searching and indexing is unchanged.** Since metadata values still have text values, the browse and search systems will work unchanged.
6. **Choice behavior decoupled.** The selection and choice mechanisms can be invoked independently (e.g. in the submit UI) of authority control.

Behavior

Interactive Submission

When collecting a value for an authority-controlled field, the interactive submission UI has to help the user choose a value from the authority set. Typically the user enters a clue or partial value and is then presented with a list of *matches* from which to choose. Each potential answer may include not only the value of the metadata field, but also some associated information that helps discriminate between identical values. For example, an authority on personal names might include title, department, age, and other details to help the user choose between two records with identical names.

An authority service must be able to provide both *search* and *browse* functions, the former returning all authority records matching a proposed value, and the latter used to populate menus of choices. (It is not always practical to browse choices, since there may be a large number of them, so some fields may only offer the search option.)

Unattended Submission

All of the batch and package-oriented submission methods are considered *unattended* since there is no provision for a user interface to get a person to make decisions. Metadata values are typically assigned by crosswalks which translate from external MD formats to DSpace's "Dublin Core"-style fields.

When the crosswalk adds a metadata value to an authority-controlled field in an Item, the authority is given a chance to test the value. It returns both an authority key and a *confidence* value, which is a measure of how well the authority key (if any) represents the metadata value. There are more details in the API section below.

The crosswalk *can* also be coded to look up an authority key itself, and supply it explicitly when setting the metadata value. In this case it determines the authority key (and confidence level) that gets set in the Item. For example, if you implement a SWORD client that consults the same name authority as DSpace for personal names, it can just pass the authority key through the encoded metadata to compatible crosswalk that puts that key directly into the appropriate Item metadata field.

Otherwise, when ingesting a raw text value into an authority controlled field, the unattended environment must be prepared for any of these conditions:

1. If the authority made a successful match, just record the metadata value with its authority code and confidence
2. If not, but the field does not *require* authority control, record the metadata value *without* any authority information.
3. If an authority value *is required* and there is a result with low confidence, record the values along with that confidence and hope it is caught and repaired in manual workflow later.
4. When an authority value *is required* but the supplied value is clearly unacceptable, signal an error that aborts the submission.

The exact handling of the various levels of failure for each kind of metadata field is a matter of policy that is ultimately decided by the crosswalk implementation. It is influenced by the confidence metric provided by the authority plugin. The authority can advise the crosswalk when to declare an error, or just accept a dubious match.

Metadata Editing

The administration UI includes a page that lets administrators edit an Item's "Dublin Core" metadata, changing values and adding fields. It must also enforce authority control on the fields for which it is configured. When an authority key is available, it is shown in the UI as a read-only text field. Fields with authority control also present a "Lookup" button which invokes a generic UI to solicit a value by taking clues to look up and displaying matches from the authority. The free-text value of the field may also be changed independently from the authority key.

Batch Metadata Editing

See Batch Metadata Editing Feature for details about this proposal. The implications are essentially the same as for unattended submission: it can apply authority keys directly when available, otherwise do (explicitly or implicitly) a lookup of values set on authority-controlled fields.

Correcting Unattended Submissions and Edits

When an unattended submission or edit leaves an authority-controlled field with a *problem*, e.g. an ambiguous or unidentified value, there are two ways to correct it:

1. Workflow states that allow metadata editing can present the same tools as the interactive submission UI.
2. If the Item is past workflow, the administrative metadata editing UI must be used.

We will consider adding an administrative UI to detect and list these metadata problems.

Display and Crosswalk

Every metadata value potentially includes values for an *authority key*, and an *authority confidence*. They are only present when the field is under authority control, and when there is an actual authority value, otherwise they are absent (e.g. in the DIM XML representation) or null.

The presentation UI can call on a generic method to get the canonical display string for an authority key, but it is welcome to interpret it in custom code to present a more detailed view. For example, one site may want to customize their Item display so a personal name appears with a link to their page on the institution's social networking site, which it obtains through the authority key.

Dissemination crosswalks will also receive the authority key so they can pass that knowledge on through OAI-PMH, exported packages, and any other dissemination vehicles.

Browse

There is a new field type in the Configurable Browse facility that has the browse mechanism index Items by their authority key values instead of the text value of the metadata field. This gives a truly authority-controlled browse UI.

Search

The support for authority control in search is very subtle:

For every search index that includes an authority-controlled metadata field, another index is automatically created just for the authority values. It is given the name of the index followed by `"authority"`. *Note that its search terms are the raw authority keys, _not* the text metadata values – those are still indexed normally.

See the UI discussion below for more details and an example.

Deliberate Omissions

Since the resources to design and implement this prototype are sharply limited it is necessary to pare it down to the features essential for our site.

No Changes to Search UI

Although search indexes are built for the authority keys of authority-controlled fields, there is no explicit UI to access these indexes. For example, they do not appear in the index list of Advanced Search in the XMLUI. This is appropriate since most users would not know how to get authority key values to search on anyway.

There *are* other mechanisms, including OpenSearch, which can easily access the authority indexes.

No *Explicit* Support in Batch Importer

The "batch" import mechanism only supports authority control implicitly, through the same backward-compatibility mechanism that attempts to assign authority values automatically on any unattended ingest. This implementation has no way to include an authority value in the DC metadata, although of course this feature can always be added later if there is enough demand and a willing developer.

Since the batch ingester has substantially been replaced by the package ingester, which is available through the LNI and SWORD as well as through command-line invocation, we *will* concentrate on adapting crosswalk plugins, and thus the package ingester, to work with authority control.

One-to-Many Model of Authority Control

The data model (as detailed below) makes the simplifying assumption that there is one canonical displayable representation for each key describing an entry in the metadata-value authority. In practice this is not always the case, e.g. in an author name authority, a single identifier can have multiple records which are considered canonical. In the oft-cited example of an author writing under pseudonyms, both "Philip Jose Farmer" and "Kilgore Trout" might be bound to the authority record for the same individual, at equal levels of confidence. DSpace relies on the plug-in implementation communication with that authority to order the choices, and in some situations (e.g. unattended submission) it must blindly choose the first.

Data Model

Relational Tables

The basic implementation only adds two columns to the MetadataValue table:

Oracle:

```
ALTER TABLE MetadataValue
    ADD ( authority VARCHAR(100),
          confidence INTEGER DEFAULT -1);
```

Postgres:

```
ALTER TABLE MetadataValue
    ADD COLUMN authority VARCHAR(100),
    ADD COLUMN confidence INTEGER DEFAULT -1;
```

These allow an *authority key* and *confidence metric* to be associated with each value. Note that some other significant state associated with the authority control of a metadata field is in the DSpace Configuration because it is a property of the software, not the data.

Some indexes will probably be needed once we get some experience with the prototype implementation to see what the query behavior is like.

Authority Key

The *key* is a text string whose interpretation is left up to the authority plugin serving that metadata field.

It is expected to have these properties:

1. The key *must not* contain any confidential information, e.g. email address or personal identification number, since it gets passed around in HTTP transactions (e.g. the browse UI).
2. Each key must correspond one-to-one with a unique record of a value for this field, in the naming authority.
   - The contents of the key have *no meaning* to DSpace, it is simply stored and passed back to the authority plugin.
   - Of course the key may be meaningful in other contexts, e.g. an ISSN or ISBN, an "info" URI, LC name identifier, etc.
3. Its size is currently limited to 100 characters, although that may prove too small if e.g. URIs are used as keys.

Confidence

An integer describing the level of "quality", or confidence, that the authority key is the correct and unique representation of the text value of the metadata field. See the API for a description of its meaningful values.

The confidence value is mainly necessary because we support setting an authority value in an *unattended* environment, so it shows how much "confidence" we should have in the authority value. The act of choosing an authority entry to match the submitted metadata value is inherently imprecise: the proffered value might match multiple entries, or none, or the operation might fail because an external resource is unavailable.

We do not want the entire Item ingestion to fail (at least, not *always*) because of what may be a minor problem in the metadata. However, we *also* do not want to let incorrect or incomplete metadata get recorded, unremarked. The solution is to add a mechanism to grade the metadata, so that we can detect problems and direct human operators to fix them. The *confidence* metric is that mechanism, upon which we can implement any policy.

Confidence Values

There are symbolic constants (and corresponding String symbolic names) for the confidence levels defined in the **Choices** class:

Icon	Confidence	Confidence value (stored in metadatavalue)	Description	Mirage 2 Glyphicon rendering

	ACCEPTED	600	This authority value has been confirmed as accurate by an interactive user or authoritative policy	thumbs-up
	UNCERTAIN	500	Authority value is singular and valid but has not been seen and accepted by a human, so its provenance is uncertain	cog
	AMBIGUOUS	400	There are multiple matching authority values of equal validity	question-sign
	NOTFOUND	300	There are no matching answers from the authority	thumbs-down
	FAILED	200	The authority encountered an internal failure in trying to match the value	warning-sign
	REJECTED	100	The authority recommends this submission be rejected	
	NOVALUE	0	No reasonable confidence value is available	ban-circle
	UNSET	-1	No confidence value has been set (default value in the DB table)	remove

The icon rendering in the XMLUI Mirage 2 theme takes place in choice-authority-control.xsl.

Separation of *Choices* from *Authority Control*

The prototype implementation has a feature that was not in the original design proposal: a *choices* mechanism that is distinct from the machinery of authority control. This is an advantage because:

1. Choices are useful outside of authority control: Some metadata fields require a value chosen from a restricted set but do not have a concept of authority keys.
2. The same choice management can also be applied to authority-controlled fields.
3. All of the configuration and UI frameworks can be shared between both authority and non-authority fields using the choice mechanism.

Metadata Fields

All configuration is determined *by MetadataField*: it is the *field* that gets declared as the object of a particular set of choices, or as authority-controlled. This ensures the field has uniform treatment throughout the DSpace platform, e.g. in browse indexing, submission, dissemination, etc.

User Interface

Public (Artifact Browser) UI

Choice control has no visible effect on the public UI, except that metadata values of choice-controlled fields will be restricted to the controlled set. This can help to clean up and normalize indexes for browsing or crosswalking/interfacing to other systems.

Authority control values are normally not visible in the UI either, although the presence of authority and confidence values has some effects:

1. Browse indexes can be configured as *authority-controlled* and this gain the benefits of authority keys.
2. In XMLUI, the "full" metadata view of an Item shows a *confidence icon* next to a metadata value with a non-empty authority key.

Submission UI

Fields configured with choice plugins will appear on the submission "Describe" pages as dictated by their chosen *presentation style*.

Fields configured as authority-controlled will also display a *Lookup* button (or *Lookup and Add* for repeatable fields), *and* a confidence icon will appear when an authority key has been determined.

Administrative UI

The "Edit Item Metadata" page is affected as follows:

1. Choice-controlled fields have either a *Lookup* button or a selector if the presentation style is `select`.
2. Authority-controlled fields furthermore include an authority key box below the field, and a confidence icon:
   a. The authority key is normally read-only, but it may be changed after clicking the "unlock" icon.
   b. If you change the authority key interactively, the confidence value becomes *ACCEPTED* since it was set by a human.
   c. A confidence icon is always shown, even when there is no authority key. This is because the confidence *is* significant; when it is *ACCEPTED* it means a human approved the authority value *even* if it is blank.

Searching

Although DSIndexer automatically builds a separate index for the authority keys of any index that contains authority-controlled metadata fields, the "Advanced Search" UIs does not allow direct access to it. Perhaps it will be added in the future.

Fortunately, the OpenSearch API lets you submit a query directly to the Lucene search engine, and *this* may include the authority-controlled indexes.

Example of Authority-Controlled Search

For this example, suppose the DC metadata field `dc.contributor.author` is authority-controlled. The search index `author` is configured by default to include the fields `dc.contributor.*`, so it effectively *inherits* authority control. Thus, there will be a separate search index `author_authority` created for the authority keys. Note that *only* the Items with authority key values are represented there, it will probably be a *subset* of the `author` index.

To search on this index, you would submit an OpenSearch query such as this to retrieve Items with an author whose authority key matches `no2004117088`:

```
http://dspace.myuni.edu/open-search/?query=author_authority:no2004117088
```

Obtaining Authority Keys

How do you get the authority key value on which to search? If it comes from an external source (e.g. the Library of Congress Naming Authority), and you happen to know the details of how it is derived, you can derive it directly from the source. Otherwise, you can use the *Browse* UI on an authority-controlled browse index to display a list of authority keys in the first-order browse list. Each value is a link whose URL is of the form:

```
http://dspace.myuni.edu/browse?type=*BROWSE_INDEX_NAME*&authority=*AUTHORITY_KEY*
```

e.g.

```
http://dspace.myuni.edu/browse?type=author&authority=no2004117088
```

So you can extract the authority key value right out of the URL.

Configuration and Customization

The Choice Management and Authority Control subsystems are just a framework. Without *configuration*, they have no effect on the operation of your DSpace site. The desired behavior for *each metadata field* is driven entirely by the DSpace Configuration properties and the nature of the chosen ChoiceAuthority plugin.

For each metadata field, you can configure:

1. ChoiceAuthority plugin, which marks the field as under Choice Management.
2. Presentation style.
3. Closed or open choices.
4. Authority controlled (choice management also required).
5. Authority value required.

Relationship with Interactive Submission Configuration

Since choice management and authority control affect the operation of the interactive submission pages, there is naturally some interdependence with that configuration as well. This table shows how the data type chosen for a metadata field affects choice and authority management:

input-type (from input-forms.xml)	'lookup' Presentation Style	'suggest' Presentation Style	'select' Presentation Style	Authority Control
onebox	yes	yes	yes	supported
twobox	yes	yes	yes	supported
textarea	yes	yes	yes	supported
name	yes	NO	NO	supported
date	NO	NO	NO	n/a
series	NO	NO	NO	not tested
dropdown	NO	NO	yes	NO
qualdrop_value	NO	NO	NO	NO
list	NO	NO	???	not tested

Other Restrictions

Plugins and "Select" Presentation Style

Not every ChoiceAuthority plugin can present the **select** presentation style. The plugin *must* be able to respond with a complete list of choices even when no query value was specified. If that is not possible (e.g. it searches a network resource with tens of thousands of journal titles), then it should *not* be configured with the "select" style.

"Name" input type

A plugin intended to manage choices for a personal-name field must be coded to expect its query value in the DSpace canonical name format, i.e. "Last, Firsts", e.g. "Doe, John", "Adams, John Quincy", "King, Martin Luther Jr.".

Configuration Properties

Choice Management Configuration

ChoiceAuthority Plugins

First, configure all your ChoiceAuthority plugins in the usual PluginManager style. The example plugins provided in the DSpace code are configured here:

```
plugin.named.org.dspace.content.authority.ChoiceAuthority = \\
  org.dspace.content.authority.SampleAuthority = Sample, \\
  org.dspace.content.authority.LCNameAuthority = LCNameAuthority, \\
  org.dspace.content.authority.SHERPARoMEOPublisher = SRPublisher, \\
  org.dspace.content.authority.SHERPARoMEOJournalTitle = SRJournalTitle, \\
  org.dspace.content.authority.SolrAuthority = SolrAuthorAuthority
```

Automatic Choice Authority from Configurable Submission value-pairs

The default configuration also includes a special self-named plugin that picks up all the *value-pairs* elements defined in your `input-forms.xml` configuration and makes them available as choice authorities (especially suitable for the **select** presentation style:

```
plugin.selfnamed.org.dspace.content.authority.ChoiceAuthority = \\
  org.dspace.content.authority.DCInputAuthority
```

Some of the ChoiceAuthority instances available in the default configuration are:

- **LCNameAuthority** - *Sample* Library of Congress (USA) name authority - NOT for serious use.
- **SRPublisher** - Journal Publisher names based on SHERPA/RoMEO database
- **SRJournalTitle** - Journal Titles based on SHERPA/RoMEO database

The `org.dspace.content.authority.DCInputAuthority` plugin picks up all of the `value-pairs` tags from the `input-forms.xml` configuratino and *automatically* creates plugin instances out of them, named by the **name** attribute they have in the config. The default DSpace config includes these choice authorities:

- common_types - List of dc.type values from `input-forms.xml`
- common_iso_languages - List of dc.language.iso values from `input-forms.xml`
- common_identifiers - List of dc.identifier.X qualifiers from `input-forms.xml`

Selecting the choice plugin

First, any authority-controlled field must *also* be configured with a source of choices. This is also defines a simple choice field:

```
choices.plugin._schema.element.qualifier_ = _plugin-name_
```

e.g.

```
choices.plugin.dc.relation.journal = SRJournalTitle
```

Selecting Choice Presentation Style

This determines the UI presentation of the choice (mainly in the interactive submission UI).

```
choices.presentation._schema.element.qualifier_ = select | suggest | lookup
```

e.g.

```
choices.presentation.dc.relation.journal = suggest
```

The available values are:

- `lookup` - User enters a proposed value and clicks a button to "look up" choices based on that value, and present a pop-up window that lets her navigate through choices.
- `suggest` - As the user types in a text-input field, a menu of suggested choices is automatically generated. It acts like the Google Suggest feature.
- `select` - Puts up a drop-down menu (or multi-pick selection box) of choices using the HTML SELECT widget.

Open or Closed Choices

Finally, the choices for a metadata field may be specified as *open* (i.e. values not included in the choices are allowed) or *closed* (restricted to the set of values offered. The default is *open*. This means that when a proposed value is *not* already in the choices, it is added as an allowable choice.

```
choices.closed._schema.element.qualifier_ = true | false
```

e.g.

```
choices.closed.dc.relation.journal = false
```

Authority Control configuration

Marking Authority Control

To declare a field as authority-controlled, just add a property like this, *in addition to* its Choices plugin declaration:

```
authority.controlled._schema.element.qualifier_ = true
```

e.g.

```
authority.controlled.dc.relation.journal = true
```

Requiring Authority Value

To further constrain an authority-controlled field so that it *must* have an authority key whenever setting a metadata value, add the property:

```
authority.required._schema.element.qualifier_ = true
```

CAUTION: Making an authority required might cause an unexpected error if that metadata field is set to a value for which the choices plugin cannot find any authority keys.

Setting Minimum Confidence

The *minimum confidence* is the lowest level of confidence at which a metadata value is included in an authority-controlled browse or search index. This may be set per-field, and there is a configurable global default.

The confidence is expressed as a symbolic value. It is case-insensitive and must be one of the following words, listed in descending order:

- accepted
- uncertain
- ambiguous
- notfound
- failed
- rejected
- novalue
- unset

For example:

```
authority.minconfidence.dc.contributor.author = accepted
```

Default Minimum Confidence

The default minimum confidence value is configurable with the property

```
authority.minconfidence
```

. The built-in default is `ACCEPTED`, but if you find this is too high, you may prefer `AMBIGUOUS` to give automatically-derived authority keys the benefit of the doubt, e.g.

```
authority.minconfidence = ambiguous
```

Example of some field configurations

```
# Simple configuration of authory-controlled author, authority not required
 choices.plugin.dc.contributor.author = LCNameAuthority
 choices.presentation.dc.contributor.author = lookup
 authority.controlled.dc.contributor.author = true
 #
 # As an example, get journal title for dc.title.alternative
 choices.plugin.dc.title.alternative = SRJournalTitle
 choices.presentation.dc.title.alternative = suggest
 #
 # This employs a select to restrict choices for dc.type field on EditItemMetadata page:
 choices.plugin.dc.type = common_types
 choices.closed.dc.type = true
 choices.presentation.dc.type = select
```

Customization

Adding ChoiceAuthority Plugin

You'll probably want to implement or adapt your own version of a `ChoiceAuthority` plugin. See the API description for more details, and consult the sample implementations in the `org.dspace.content.authority` package.

Customizing Look + Feel

XMLUI

For the XMLUI, the prototype includes sample styles and images in the `Reference` theme.

Authority Confidence

You can control the images shown for various authority confidence levels with style tags such as `img.ds-authority-confidence.cf-`*NAME*, where *NAME* is the *symbolic name* of the confidence level, such as **accepted** (see above for the list). For example, this displays a thumbs-up icon to mark a human-approved level of confidence in the authority value:

```
img.ds-authority-confidence.cf-accepted
  \\{ background: transparent url(../images/confidence/6-thumb2.gif); \\}
```

Debugging Authority Values

You can turn on the display of authority-value fields, ideally just for debugging since it clutters the display. Adjust the value of the `display` style to `inline` to see authority value fields on Submission UI forms. (Note that the Edit Item Metadata page already displays authority values.)

```
input.ds-authority-value \\{ display: none; \\}
```

Suggest / Autocomplete

The prototype CSS also includes some styles for the Scriptaculous JavaScript autocomplete feature as well. These lines should be copied (and customize as necessary) into your theme's CSS:

```
div.autocomplete
 div.autocomplete ul
 div.autocomplete ul li.selected
 div.autocomplete ul li
 div.autocomplete ul li span.value
```

JSPUI

For customization, examine the contents of these pages in the webapp:

```
/tools/lookup.jsp
 /tools/edit-item-form.jsp
 /submit/edit-metadata.jsp
```

"Lookup" page

The popup page generated for the *lookup* presentation style is created by the underlying UI mechanism in both JSPUI and XMLUI. All of the usual customization and styling tricks for that UI thus apply to it.

Prototype API

New Classes

Metadata Authority Control class

Actual authority control is mainly a matter of the `MetadataAuthorityManager` broker class, which interprets the DSpace configuration and reports the authority status of a field. That's about all it does.

Choice Authority Manager

The *choices framework* consists of a `ChoiceAuthorityManager` class which serves as a broker and accesses the configuration, and individual plugins implementing the
`ChoiceAuthority` interface. Each plugin represents a *choice authority*, which is a source of value options or *choices*. See the prototype code for details.

Choice Authority plugin

These are the significant methods in the `ChoiceAuthority` interface:

getMatches

Returns the set of choices *matching*, i.e. possibly relevant to, a proposed (and maybe partial) metadata value.

The exact requirements and expectations of this method's implementation depend on how the fields that call it are configured: if the *suggest* UI presentation is employed, it will get called with partial values. If only the *lookup* presentation is called, it will see complete values. Also, the *suggest* mechanism is not usually capable of taking "paged" result sets (i.e. where *start* is greater than 0).

Note that getMatches() is given a *collection* argument, which contains the owning Collection of the Item (or SubmissionItem) for which a metadata value is being assigned. This is intended to give the plugin some *context* to adjust its criteria for assembling a set of choices. For example, a personal-name choice authority may restrict its search to members of a certain department if the indicated collection is only intended for works by members of that department.

The *collection* is supplied as a database ID in order to save the overhead and database access which would be required to create a DSpaceObject instance – on the assumption that it is not always going to be used, and in fact probably pretty rarely, *and* that the speed of a choice request is very important in the interactive context, the expense of querying the collection is deferred so it is only incurred by those implementations which really need it.

```
public Choices getMatches(String text, int collection, int start, int limit, String locale);
```

getBestMatch

Gets the single "best" match (if any) of a value in the authority to the given user value. This is also expected to return a meaningful "confidence" expressing the the circumstances of this match, i.e. if it is ambiguous or not.

This call is typically used only in non-interactive metadata ingests and other contexts where there is no opportunity for an interactive agent to choose from among options.

Note that getBestMatch() is given a *collection* argument, just like the collection context given to getMatches() – see above for details.

```
public Choices getBestMatch(String text, int collection, String locale);
```

getLabel

Get the canonical, human-readable *label* (i.e. short descriptive text) corresponding to the *authority key* of a value. This is *only* called for fields defined as authority-controlled; a choice plugin that is never used for authority-controlled fields does not need to implement this.

```
public String getLabel(String key, String locale);
```

API Changes

Changes to `Item`

Add methods that take authority-key and confidence arguments:

```
public void addMetadata(String schema, String element, String qualifier, String lang, String value, String
authority, int confidence)
 public void addMetadata(String schema, String element, String qualifier, String lang, String[] values, String
authorities(), int confidences([])
```

IMPORTANT NOTE on Backward Compatibility

The old `addMetadata` methods are still available, of course, although they have a new failure mode. Whent the field is authority-controlled they will call the field's configured `ChoiceAuthority`'s `getBestMatch()` method to generate an authority key and confidence value. If the field is configured to *require* an authority key, an exception is thrown if `getBestMatch()` does't return one.

Changes to `Metadatum`

Add fields:

```
public String authority;
 public int confidence;
```

Changes to `MetaDataValue`

```
public String getAuthority()
 public void setAuthority(String value)
 public int getConfidence()
 public void setConfidence(int value)
```

Changes to `DIM` XML "schema"

The DIM Field element gains two new attributes to represent the authority and confidence (when available) of DC metadata values, e.g.

```
<dim:field schema="dc" element="contributor" qualifier="author" *authority="n79-21164" confidence="6"*>
 Mark Twain
 </dim:field>
```

Changes to DRI Schema

The XMLUI's DRI schema has some attributes added to support choice options and authority control in input fields and their values. Since the pages in the Item Submission and Item Edit Metadata UIs effectively round-trip all of the Item's DC metadata values through a Web page, it is essential for the page's fields to preserve any authority key and confidence values as well.

DRI `params` Element

The `params` subelement of `field` gets some new attributes:

- `authorityControlled` - Boolean, true if the field is authority-controlled.
- {{authorityRequired}} - Boolean, true if the field requires an authority key value.
- `choices` - Name of the metadata field whose choice configuration to use.
- `choicesPresentation` - Type of choices presentation style to use, either *suggest* or *select*
- `choicesClosed` - Boolean, true if choice is configured as *closed*, i.e. no values outside of presented choices are to be allowed.

The `value` subelement of `instance` and some `field` (e.g. text, textarea) elements gets some new attributes:

- `type='authority'` - This new keyword value of the *type* attribute means the value is an authority-key value.
- `confidence` - The given confidence number applies to this element, which should also have a *type* of `"authority"`.

New Classes

In the `org.dspace.content.authority` Package

- `Choice` - record class for a single choice value
- `Choices` - record class for result of choices query
- `ChoiceAuthority` - interface of choice authority plugin
- `ChoiceAuthorityManager` - choices factory and access
- `MetadataAuthorityManager` - metadata authority control factory

XML User Interface changes

- New `/choices/`*field*URL added to sitemap to retrieve choices data as XML, for AJAX browser scripting.
  - Implemented by `org.dspace.app.xmlui.cocoon.AJAXMenuGenerator`
- Show icons depicting confidence level of authority-controlled metadata values, in long Item view and MD edit pages.
- Option to show authority values in submission UI, for debugging, see `input.ds-authority-value` stanza in CSS file.

XMLUI Sample Implementation

Any Theme for a DSpace using the Choice or Authority Control features *must* include the CSS declarations prototyped in the **Reference** theme. The corresponding images must also be available in your theme. All of the other elements (JavaScript, translations, etc) are implemented below the theme layer.