

2

Working with the crawler UI

This chapter covers

- Learning to navigate the crawler UI
- Understanding the conventions of the crawler UI
- Using the UI to perform a web crawl

In this chapter you will learn about the crawler UI itself in much greater depth than in Chapter 1. We will explore all of its features in considerable detail, and we'll even learn a little bit about how it is put together. We will do this in the context of performing a simple web crawl.

2.1 *UI architecture*

Up until now, Bert has been more than happy to blindly follow your suggestions and instructions as to how to control ManifoldCF. But now that he is growing comfortable with the software, he wants more details. Specifically, he wants to learn more about the crawler's user interface. He wants to know "what all the buttons and links do".

Bert's request is perfectly reasonable, because underlying all the screens, buttons, and links in the UI is an understanding of how ManifoldCF's world is put together. So in walking through them, we're going to learn a lot about how to *think* about ManifoldCF. But in order to fully exercise the UI's capabilities and understand its conventions, we're going to have to move beyond the very simple example used in the first chapter. We're going to set up and perform a web crawl. We need to do this because every connector makes use of different aspects of the core framework, and the web connector happens to use more of that framework than any other connector.

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

Even so, it is not possible to present a complete walkthrough of the UI, since each connector contributes its own components to that UI. We'll discuss this in more detail later. But first, let's revisit the crawler UI's basic structure, in order to understand how it actually works.

2.1.1 *UI conventions*

The ManifoldCF user interface was carefully constructed to be cleanly extendable. Thus it adheres to a set of basic conventions, or rules. When you write connectors of your own, you will need to know all about these conventions and rules, since each connector includes its own UI. In this section we'll summarize these rules at a high level. If you need more detail, that may be found in Chapter 8.

TECHNOLOGY

First, it's probably obvious that the ManifoldCF crawler UI is written with the most basic of building blocks: HTML forms. Its use of Javascript is not even terribly complex. With all the cool Ajax UI's out in the world today, this may seem like a manifestly un-fun way to do it.

In this case, though, simpler was deemed better. For one thing, every connector author will need to write what amounts to a chunk of the UI on their own, and people have wildly different experience levels with the technologies that might have been used. Even more importantly, it is relatively straightforward to write automated tests for a form-based user interface, using basic browser emulation tools.

UI TRANSACTION MODEL

The ManifoldCF UI, like many other UI's, uses a transactional model for editing the entities it manipulates. Every time you edit a connection definition or a job definition, the crawler UI does the following:

1. Load the entity from the database
2. Present all or part of the entity's information to the user, preserving the rest in hidden form elements
3. Post the form data, repeatedly, as you click on the various tabs
4. On the "Save" post, save the entity's information back to the database

Since the entity is not modified until the `Save` button is clicked, changes will be discarded if you either cancel the editing transaction, or navigate away from the page. The changes are effectively made to be atomic by virtue of the way the UI is constructed. Once again, this is not an unusual paradigm for HTML form-based UI's.

What **is** somewhat more unusual is that no attempt is made in ManifoldCF at this time to prevent more than one person from changing the same entity at the same time. Instead, if this happens, one person's changes will unfortunately be overwritten by the other person's changes, with no warning of any kind. The reason for this omission is lack of necessity:

heretofore it has been extremely rare for more than one person to be working with ManifoldCF's UI at the same time.

TABS

When you edit any entity in the ManifoldCF UI, it's a fair assumption that all of the data corresponding to that entity will not fit comfortably on a single screen. Indeed, given that some components of the entity (such as a connection definition's configuration information, or a job definition's document specification) are hierarchical structures that can be arbitrarily complicated, finding a good general way to present and edit complex entity information is not a trivial task at all. ManifoldCF does its best to find a solution that is both usable as well as sufficiently powerful: it organizes the entity's data into *tabs*.

As you have seen, this organization has many benefits. Not only can data be organized in a logical fashion, but it is also possible to every connector to contribute its own specific set of tabs to the UI, without disrupting tabs contributed by ManifoldCF itself, or by other connectors. But one side effect of all this flexibility is that, as I've already stated, a complete walkthrough of the user interface is not possible, because the tabs will vary considerably from connector to connector. The best one can do is to describe the core set of tabs for each entity.

2.1.2 Navigation menu

You've seen the navigation menu before. Referring back to figure 1.7, we can see the full menu on the left. What do all of these menu selections actually do? The menu items and their functions are listed in table 2.1.

Table 2.1: The crawler UI navigation menu options

Item	Function
List Output Connections	Presents a list of the current output connection definitions, allowing editing, deletion, inspection, and creation of new output connection definitions
List Authority Groups	Presents a list of the current authority groups, allowing editing and deletion
List User Mapping Connections	Presents a list of the current user mapping connection definitions, allowing editing, deletion, inspection, and creation of new connection definitions
List Authority Connections	Presents a list of the current authority connection definitions, allowing editing, deletion, inspection, and creation of new authority connection definitions
List Repository	Presents a list of the current repository connection definitions, allowing

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

Connections	editing, deletion, inspection, and creation of new repository connection definitions
List All Jobs	Presents a list of the current job definitions, allowing editing, deletion, inspection, and creation of new job definitions
Status and Job Management	Presents a list of the operational status of all the current jobs, allowing various actions such as job start, abort, restart, pause, and resume
Document Status	Examines the job queue for a specific set of documents and for a specific set of jobs
Queue Status	Summarizes the job queue for a specific set of documents and for a specific set of jobs, by counting
Simple History	Examines the history for a repository connection, for a specific time interval, set of entities, set of result codes, and set of activities
Maximum Activity	Finds the maximum number of events in the history within a time window of a specified size
Maximum Bandwidth	Counts the maximum number of bytes in the history within a time window of a specified size
Result Histogram	Groups history entities and results according to a regular expression
Help	Links to the online ManifoldCF end-user documentation

In the following sections, we're going to visit or re-visit every page of the ManifoldCF user interface, with the goal in mind of learning the UI's capabilities. While we are at it, we will set up and perform a simple web crawl as our chapter example. By the time we are done, you should have a pretty solid idea of how to do nearly everything you might need to, using the Crawler UI.

2.2 Connection definition management

Setting up connection definitions is a fundamental function of the ManifoldCF Crawler UI. Each kind of connection definition has its own link in the navigation area, but you will quickly notice that each kind of connection definition has more in common from a UI perspective than not. So it makes sense to think of all three kinds of connection definition together.

First, we'll revisit output connection definitions, to clarify some UI aspects we glossed over in Chapter 1. Afterwards, we'll briefly have a look at authority connection definitions, which we haven't yet had occasion to work with, and then revisit repository connection definitions, with the goal in mind of setting up a web crawl, which will be our example for this chapter.

2.2.1 Managing output connection definitions

When you click on the `List Output Connections` link on the navigation menu, you enter the area of the UI which manages output connection definitions. We will need an output connection definition of some kind in order to demonstrate web crawling. But we can certainly make do with the same output connection definition that you have already defined and used in Chapter 1, which uses the null output connector. Nevertheless, let's revisit the UI pages we've already seen, so we can discuss them in greater depth.

OUTPUT CONNECTION DEFINITION LIST FIELDS

The presented list of output connection definitions provides a little general information about each connection definition. While incomplete, it helps you keep track of exactly which connection definition is which. Figure 1.8 shows the UI page in question. The fields are described in table 2.2.

Table 2.2: Output connection definition list fields

Field	Meaning
Name	The name of the connection definition, which is unique
Description	A description of the connection definition, which hopefully is helpful, but does not need to be unique
Connection Type	The underlying connector used for the connection definition
Max	The maximum number of instances for this connection that ManifoldCF will permit to exist at one time

STANDARD OUTPUT CONNECTION DEFINITION TABS

The three tabs that make up the standard output connection definition tab set are the `Name` tab, the `Type` tab, and the `Throttling` tab. These tabs set the basic parameters of every output connection definition. It's also no accident that these common tabs are where the information that makes up the columns of the list of output connection definitions comes from.

THE NAME TAB

Refer to figure 1.9 for a screen shot of the `Name` tab. This tab is used solely for setting a connection definition's name and description. As mentioned before, the name given must be unique among output connection definitions, while the description can be anything, although you will benefit enormously by choosing a description that is meaningful and helpful. The name field can contain any Unicode character, but is limited to 32 characters in total. The

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

description field also may consist of any Unicode characters, but can be up to 255 characters in length.

THE TYPE TAB

We've encountered the output connection definition `Type` tab before, in Figure 1.11. Use this tab to select the output connection type (which maps to the connector class by way of a database table). If the connection type you are looking for is not in the dropdown, it is because your output connector has not been *registered*, and you will need to take care of that problem first.

THE THROTTLING TAB

Figure 2.1 shows what the Throttling tab looks like.

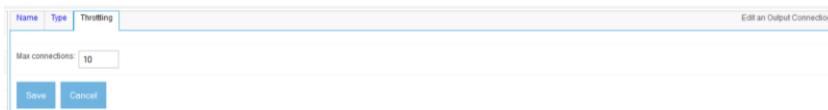


Figure 2.1 An output connection definition Throttling tab. Here you can set the maximum number of output connection instances in existence at any given time.

For an output connection definition, all you can do with this tab is to set the `Max Connections` connection parameter. This parameter limits the number of active, connected instances of the underlying connector class that have the same exact configuration. For example, you may have two Solr indexes, and a connection definition for each one. In this case, ManifoldCF will keep two distinct pools of connected Solr connection instances, and apply the corresponding limit to the number of connection instances managed for each pool.

Note: Versions of ManifoldCF prior to 1.5 had a separate pool of connection instances for each JVM that was part of the ManifoldCF cluster. Starting with ManifoldCF 1.5, though, the total number of connection instances is distributed dynamically across all cluster members, and a hard limit is enforced regardless of the number of JVMs involved.

So, what should you set this parameter to? In part, the answer depends on the semantics of the target system. If each target system connection has a cost, or there is a limit of some kind in the target system for the number of connections to it, then you will want to control the number of instances based on that constraint. If there is no need for limits of any kind, you may safely set this parameter as high as the number of working threads (50 per ManifoldCF Agents process, by default). Setting the parameter higher than that does no harm but has little benefit, since there is a limit to the number of connection instances ManifoldCF can use at any given time in any case.

Table 2.3 describes the target system constraints for the output connectors currently part of ManifoldCF.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:
<http://www.manning-sandbox.com/forum.jspa?forumID=727>

Table 2.3: Output connector connection count constraints

Connector	Constraint
ElasticSearch	Each target system connection will use an ElasticSearch listener thread, and the number of such threads is limited, so pick a value consistent with your ES setup
File system	No effective limit (other than system file handles)
HDFS	Look at your Hadoop File System setup for any limits, and set connector limits accordingly
OpenSearchServer	Each target system connection will use an OpenSearchServer listener thread, and the number of such threads is limited, so pick a value consistent with your OpenSearchServer setup
Solr	Each target system connection will use a Solr jetty or tomcat thread, and the number of such threads is limited, so pick a value consistent with your Solr setup
GTS	The target system only allows a maximum of 10 connections
Null	No limit

OUTPUT CONNECTION STATUS

Any time you finish editing an output connection definition, or if you click the `View` link from the output connection definition list, you will see a connection definition information and status summary screen, similar to figure 1.13. This screen has two primary purposes: first, letting you see what you actually have set up for your connection definition parameters, and second, showing you the status of a connection created with your connection definition. The connection status usually represents the results of an actual interaction with the target system. It is designed to validate the connection definition parameters you've described. However, it can also be very helpful in diagnosing transient connection problems that might arise (e.g. expired credentials, or a crashed Solr instance). This makes the connection definition information and status summary screen perhaps the best invention since the Chinese food buffet, so please make it a point to visit this screen as part of your diagnostic bag of tricks.

In addition to connection configuration information and status, this screen also has two buttons you can click that affect information that ManifoldCF retains that pertains to the output connection. The first button, `Re-index all associated documents`, will make ManifoldCF re-index all documents that were previously indexed, when the associated jobs are run the next time. You would need to click this button only if you had changed the configuration on the target system – perhaps, for example, changing what metadata fields exist. The second button, `Remove all associated records`, is somewhat similar, but

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

would be appropriate if you had completely deleted the index on the target system and wanted to start over. These buttons are helpful mainly in a development situation, and should be used with caution in a production system.

2.2.2 Managing authority groups

Astute readers will note that we've briefly seen a reference to authority groups back in Chapter 1. Specifically, when we defined a repository connection, we had the option of specifying an authority group to secure documents crawled by that connection.

This is because the purpose of an authority group in ManifoldCF represent a set of authority connections which collaborate to furnish security for one or more repository connection. More than one authority may participate in providing that security, because some repositories (such as SharePoint with Claims-based authorization) have a distributed and extendable authorization model, where a document can be secured by (to use the SharePoint example) native SharePoint, or Active Directory, or any other system for which an integrator has written the appropriate plugin. We'll discuss this in more detail in Chapter 4.

For now, let's walk through the UI, to learn what its capabilities are.

AUTHORITY GROUP LIST FIELDS

Clicking on the navigation menu link `List Authority Groups`, we arrive at a UI page as seen in figure 2.2.



Figure 2.2 The list of authority groups.

The fields displayed are simple; just a name and a description. That's because, other than being the link between authority connections and repository connections, an authority group really does very little.

THE NAME TAB

The only editing tab an Authority Group has is the `Name` tab. This tab simply allows you to set or edit the name and description of the Authority Group. See figure 2.3.

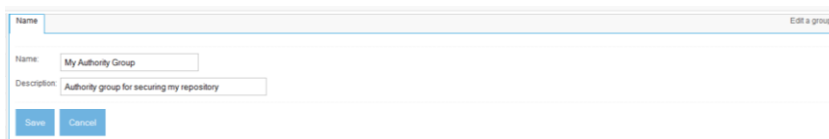


Figure 2.3 The authority group Name tab.

AUTHORITY GROUP VIEW PAGE

When you click the `Save` button, the Authority Group view page is displayed. This page simply summarized the name and description information you just entered. See figure 2.4.

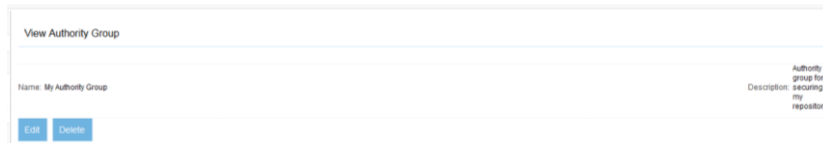


Figure 2.4 Viewing an authority group.

2.2.3 Managing user mapping definitions

The `List User Mapping Connections` link in the navigation menu brings you to the UI area where user mapping connections are managed.

We have not yet had any occasion to encounter a user mapping connection as of yet. User mapping connection definitions are meant to be used to transform user names from their starting value (which is outside of ManifoldCF) into a form recognizable by an authority connection. This is supposed to be done either via some simple textual rules, or by lookup of the mapping in some back-end system. User mapping connections can be chained together by means of a “prerequisite” mechanism, meaning that you can specify that before ManifoldCF invokes your user mapping transformation, it will invoke a prerequisite transformation first. When you specify an authority connection definition, you can likewise specify a prerequisite user mapping transformation.

But we are getting ahead of ourselves. Let’s start by having a look at the UI used to define user mapping connection definitions.

USER MAPPING CONNECTION DEFINITION LIST FIELDS

Clicking on the navigation menu link `List User Mapping Connections`, we see a UI page like the one in figure 2.5.

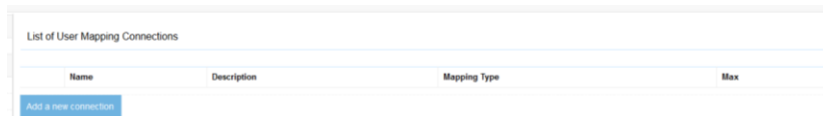


Figure 2.5 A list of the current user mapping connections.

The fields displayed for each User Mapping Connection Definition are explained below.

Table 2.4: User mapping connection list fields

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

Field	Meaning
Name	The name of the user mapping connection definition, which must be unique
Description	A description of the connection definition, which hopefully is helpful, but does not need to be unique
Mapping Type	The underlying mapping connector used for the connection definition
Max	The maximum number of connection instances for this mapping connection definition that ManifoldCF will permit to exist at one time

STANDARD USER MAPPING CONNECTION DEFINITION TABS

The four tabs that make up the standard user mapping connection definition tab set are the `Name` tab, the `Type` tab, the `Prerequisites` tab, and the `Throttling` tab. These tabs set the basic parameters of every user mapping connection definition.

THE NAME TAB

The authority connection definition `Name` tab is quite similar to the `Name` tab for output connection definitions or repository connection definitions. Its purpose is similar as well: to set the connection definition's name and description. The name can be any Unicode string, must be unique among authority connection definitions, and is limited to 32 characters. The description is limited to 255 characters. See figure 2.6 for a screen shot of this tab in the context of a user mapping connection.

Figure 2.6 The Name tab of a user mapping connection.

THE TYPE TAB

Like the output connection definition `Type` tab, the user mapping connection definition `Type` tab allows you to select which connection type to use for the connection definition. In this case, though, you will be selecting from among the user mapping connectors that have been registered with the framework. That list will depend, then, on how your version of ManifoldCF has been set up. But, out of the box, there's only one of these – a mapper which does not actually communicate with any external system at all, and simply uses regular expressions to perform user name mapping. See figure 2.7.

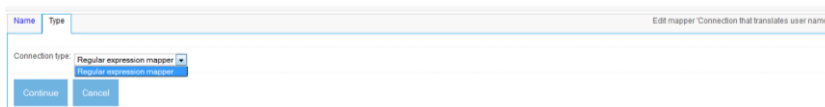
The screenshot shows a web form with a tabbed interface. The 'Type' tab is selected. At the top, there are fields for 'Name' and 'Type'. Below these, a dropdown menu for 'Connection type' is open, showing 'Regular expression mapper' as the selected option and 'Default authentication mapper' as an alternative. At the bottom of the form are 'Continue' and 'Cancel' buttons. A small text label 'Edit mapper: Connection that translates user name' is visible in the top right corner.

Figure 2.7 The Type tab of a user mapping connection. ManifoldCF only comes with one kind of user mapping, by default.

THE PREREQUISITES TAB

Wouldn't it be great if you could chain together multiple user mapping steps, some of which may include consultation with external systems, and some which simply manipulate the text of the user name? Well, in fact, you can do precisely that. Each user mapping connection definition (and each authority connection definition as well) lets you specify a user mapper that must be executed beforehand. In this way, you can define a series of user name mappings that take place before an authority connection receives it.

In our example, however, we're only defining one mapper, so there's nothing else we can specify except "No Prerequisites". See figure 2.8.

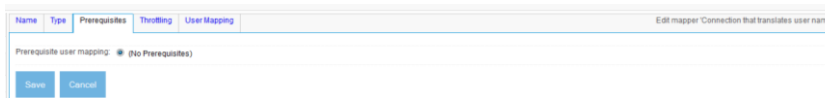
The screenshot shows the 'Prerequisites' tab of the user mapping connection form. The 'Prerequisite user mapping' field contains the text '(No Prerequisites)'. There are 'Save' and 'Cancel' buttons at the bottom. The 'Edit mapper: Connection that translates user name' label is also present in the top right.

Figure 2.8 The User Mapping Connection's Prerequisites tab. If there were any available user mapping prerequisites, you could select from them here.

THE THROTTLING TAB

Figure 2.9 shows the user mapping connection definition `Throttling` tab.

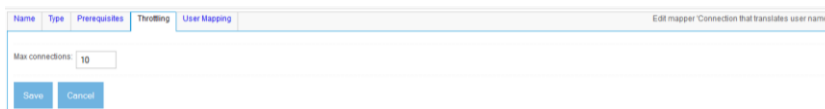
The screenshot shows the 'Throttling' tab of the user mapping connection form. It features a 'Max connections' input field with the value '10'. There are 'Save' and 'Cancel' buttons at the bottom. The 'Edit mapper: Connection that translates user name' label is visible in the top right.

Figure 2.9 The User Mapping Connection Throttling tab. Select a connection limit appropriate for the system you are connecting to, if any.

Similar to output connection definitions, the `Throttling` tab allows you to set a limit on the number of active, connected instances of the underlying connector class that use the user mapping connection definition's configuration. This might be important if, for example, you had a user mapping connection which communicated to a third-party system, such as LDAP, to look up the appropriate user name to use for a user in a specific repository. ManifoldCF will count each connection instance across all cluster members and allocation them dynamically, guaranteeing that the limit you state here is not exceeded.

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

For the regular expression user mapping connection type included with ManifoldCF, it really doesn't matter what value you set, because it does not communicate with a limited back-end system. It is only important that it does not become a bottleneck. So, you'd want to insure that there's at least one instance for each thread that is handling mapping requests. By default, the number of mapping threads is set to 10, so if you did not change that property in `properties.xml`, you would want to include at least that number for every instance of the authority service you were running in your cluster. See table 2.5 for a summary.

Table 2.5: User mapping connector connection count constraints

Connector	Constraint
Regular-expression mapper	No limit

USER MAPPING CONNECTION STATUS

Any time you finish editing a user mapping connection definition, or if you click the [View](#) link from the user mapping connection definition list, you will see a connection definition information and status summary screen, similar to figure 2.10.

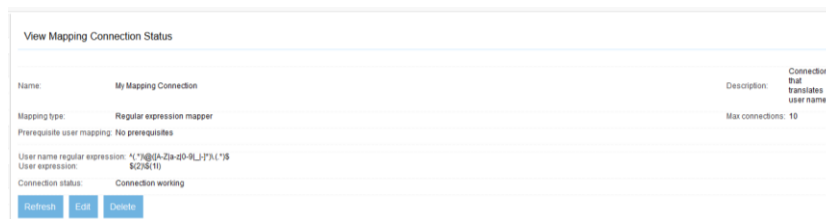


Figure 2.10 Viewing the User Mapping Connection we've just created. The default regular expression, which we didn't edit, maps "user@domain" to "domain\user".

This page includes information about the connection definition's configuration. It also includes a very useful assessment of whether the connection definition is in fact working properly in your environment (the "connection status" field).

2.2.4 Managing authority connection definitions

The [List Authority Connections](#) link in the navigation menu brings you to the UI area where authority connection definitions are managed.

We have not yet had occasion to set up an authority in this book. In part this is because the simple crawls we've done so far had no need for security. The web crawl we are setting

up now is no different. No authority connection definition will need to be defined for it. We won't need to actually define one until Chapter 4.

However, it is probably a good idea to at least walk through the authority connection definition UI pages. It's a little difficult to do this properly, because ManifoldCF currently has only one non-proprietary authority connector: the Active Directory connector, which requires a Windows domain controller around that is configured for Active Directory for it to function.

If you have access to a Windows domain controller, you should easily be able to follow along with the screen shots included here. If not, then I'm afraid you are just going to have to use your imagination.

AUTHORITY CONNECTION DEFINITION LIST FIELDS

Clicking on the navigation menu link `List Authority Connections`, we arrive at a UI page as captured in figure 2.11.

Name	Description	Authority Type	Max
Add a new connection.			

Figure 2.11 A list of authority connections. This screen is directly analogous to the list of output connections or the list of repository connections.

The fields displayed in the authority connection definition list are described below.

Table 2.6: Authority connection list fields

Field	Meaning
Name	The name of the authority connection definition, which must be unique
Description	A description of the connection definition, which hopefully is helpful, but does not need to be unique
Authority Type	The underlying authority connector used for the connection definition
Max	The maximum number of connection instances for this authority connection definition that ManifoldCF will permit to exist at one time

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

STANDARD AUTHORITY CONNECTION DEFINITION TABS

The four tabs that make up the standard authority connection definition tab set are the `Name` tab, the `Type` tab, the `Prerequisites` tab, and the `Throttling` tab. These tabs set the basic parameters of every authority connection definition.

THE NAME TAB

The authority connection definition `Name` tab is quite similar to the `Name` tab for output connection definitions or repository connection definitions. Its purpose is similar as well: to set the connection definition's name and description. The name can be any Unicode string, must be unique among authority connection definitions, and is limited to 32 characters. The description is limited to 255 characters. See figure 2.12 for a screen shot of this tab in the context of an authority connection.

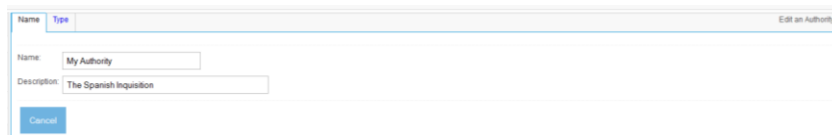
The screenshot shows a web interface for editing an authority. At the top, there are two tabs: 'Name' (selected) and 'Type'. To the right of the tabs is a link that says 'Edit an Authority'. Below the tabs, there are two text input fields. The first is labeled 'Name:' and contains the text 'My Authority'. The second is labeled 'Description:' and contains the text 'The Spanish Inquisition'. At the bottom left of the form is a blue button labeled 'Cancel'.

Figure 2.12 An authority's `Name` tab. The name of the authority connection definition is unique, and is also used to define a security space. Document access tokens sent to a search engine by ManifoldCF will include this name, so choose wisely.

THE TYPE TAB

Like the output connection definition `Type` tab, the authority connection definition `Type` tab allows you to select which connection type to use for the connection definition. In this case, though, you will be selecting from among the authority connectors that have been registered with the framework. That list will depend, then, on how your version of ManifoldCF has been set up. See figure 2.13.

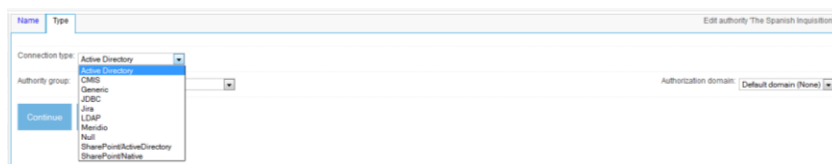
The screenshot shows the 'Type' tab of the authority connection definition interface. The title bar says 'Edit authority "The Spanish Inquisition"'. There are three main sections. The first is 'Connection type:' with a dropdown menu showing 'Active Directory' selected. The second is 'Authority group:' with a dropdown menu showing 'CMS' selected. The third is 'Authorization domain:' with a dropdown menu showing 'Default domain (None)'. On the left side, there is a list of available connectors: 'Active Directory', 'CMS', 'Generic', 'JDBC', 'Jira', 'LDAP', 'Mando', 'Mail', 'SharePointActiveDirectory', and 'SharePointNative'. A blue 'Continue' button is located at the bottom left.

Figure 2.13 The authority connection definition `Type` tab. Here, select what authority connector to use. The list will differ depending on what connectors you've built and registered with the framework.

Let's select the Active Directory connection type. This connection type is appropriate for use for security for many Windows-based repository connection types, such as FileNet, Meridio, Windows Shares, and the SharePoint connector (when operating in legacy mode).

On this page, you will also need to select the authority group that the authority connection should belong to. Indeed, if you haven't yet set up an authority group, ManifoldCF will not permit you to define an authority connection. But if you need to, you can change the authority group that the authority connection is associated with in the future.

Finally, on the right hand side you will note that you can select an "Authorization Domain" for this connection. But what, exactly, is an authorization domain? Simply put, ManifoldCF recognizes that there may be multiple ways of describing a single user. For example, perhaps in addition to an Active Directory user name, the same user may also have a different Documentum user name, or a Facebook user name, etc. In the ManifoldCF world, we call these "authorization domains", and one or more of these user name/authorization domain pairs are expected to be given to ManifoldCF whenever it is asked to provide authorization for content. Authorization domains are registered with ManifoldCF, much like connection types are registered, so the authorization domains that are available for use with an authority connection can vary from system to system. But all systems include the default authorization domain, which is what I've chosen for our Active Directory authority example. We'll go into this in more detail in Chapter 4.

THE PREREQUISITES TAB

The `Prerequisites` tab, which I've already mentioned, allows you to optionally specify a chain of user mapping connections that must take place before your authority connection is called. See figure 2.14.

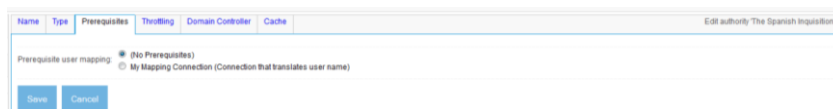


Figure 2.14 We have two choices for the user mapping connection to use for this authority: the one we defined earlier, or none at all. We'll pick the latter.

THE THROTTLING TAB

Figure 2.15 shows the authority connection definition `Throttling` tab.

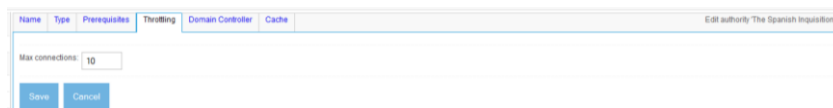


Figure 2.15 An authority connection definition's `Throttling` tab. Here you can set the maximum number of

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

connection instances corresponding to the authority connection definition that are in existence at any one time.

Similar to output connection definitions, the `Throttling` tab allows you to set a limit on the number of active, connected instances of the underlying connector class that use the authority connection definition's configuration. For example, you may have two Active Directory domains, and a connection definition for each one. ManifoldCF will keep two distinct pools of connected Active Directory connection instances, and apply the appropriate limit to the number of connection instances in each one.

Once again, what should you set this parameter to? In part, the answer depends on the semantics of the authority being connected to. If each target system connection has a cost, or there is a limit of some kind in the target system for the number of connections to it, then you will want to control the number of connection instances based on that constraint. Similarly, if each open socket connection to your authority uses up a license, you will run out of licenses very quickly if you don't set a limit. The limit applies across all cluster members, so at least you don't have to worry about that. But, if you don't want the number of connection instances to be a bottleneck, you do need to consider how many threads there may be running to fulfill authorization requests across your cluster. By default, per authority service instance, 10 threads are allocated, a number which you can change in your cluster's `properties.xml` file.

Table 2.7 lists the authority connectors currently available with ManifoldCF, and the constraints for connections for each.

Table 2.7: Authority connection constraints

Connector	Constraint
Active Directory	Windows has a tendency to behave poorly when there are too many connections to a domain controller that are active at once; consider limiting overall connections to 2-5
CMIS	The CMIS authority does not need to connect via CMIS, so there are no applicable constraints
Generic (SOAP)	Limited by the back-end system receiving the connector's SOAP requests
JDBC	Limited by the number of database connection handles available on the target database
Jira	Limited by the number of threads running on the Jira server for servicing REST API requests
LDAP	Limited by the number of threads running on the LDAP server which are servicing LDAP requests

Meridio	Windows IIS Meridio instance configuration limit
Null authority	No applicable constraints
SharePoint/ActiveDirectory	Same issue as for the Active Directory authority – consider limiting overall connections to 2-5
SharePoint/Native	Windows IIS SharePoint instance configuration limit

AUTHORITY CONNECTION STATUS

Any time you finish editing an authority connection definition, or if you click the `View` link from the authority connection definition list, you will see a connection definition information and status summary screen, similar to figure 2.16.

View Authority Connection Status

Name: My Authority Description: The Spanish Inquisition

Authority type: Active Directory Max connections: 10

Authority group: My Authority Group Authentication domain:

Prerequisite user mapping: No prerequisites

Domain Controllers:	Name	Domain controller name	Domain suffix	Administrative user name	Administrative password	Authentication	Login name AD attribute
				admin	-----	oauth2-actv-00000	username

Cache lifetime: 1 minutes
Cache LRU size: 1000

Connection status: Threw exception: 'Couldn't communicate with domain controller 'localhost: localhost:389'

[Refresh](#) [Edit](#) [Delete](#)

Figure 2.16 The view page for an authority connection definition. This connection definition has a problem, as you can see if you look at the “Connection status” line.

This screen has two important purposes: first, letting you see what you actually have set up for your connection definition parameters, and second, showing you the status of a connection created using those parameters. The connection status usually represents the results of an actual interaction with the target system. It is designed to validate the connection definition parameters you’ve described. However, as is the case for the output connection definition viewing and status screen, it can also be very helpful in diagnosing transient connection problems that might arise (e.g. invalid or expired credentials, or a crashed domain controller). This makes the connection definition information and status summary screen very useful for diagnostics.

2.2.5 Managing repository connection definitions

Clicking the `List Repository Connections` link in the navigation menu will bring you to the area of the UI dedicated to managing repository connection definitions. Here we are

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

going to define a web connection definition, which will involve going through both the standard tabs of a repository connection definition in depth, as well as learning something about the web connector's configuration tabs as well.

REPOSITORY CONNECTION DEFINITION LIST FIELDS

The list of repository connection definitions provides some information about each connection definition, although it is by no means complete. If you recall figure 1.14 from Chapter 1, you can see all of the fields described in table 2.8.

Table 2.8: Repository connection definition list fields

Field	Meaning
Name	The name of the connection definition, which is unique
Description	A description for the connection definition, which is supposed to be helpful but may not be unique
Connection Type	The underlying connector associated with the connection definition
Authority	The authority connection definition that will be used to protect content indexed using this connection definition
Max	The maximum number of instances of a connection created using this definition that ManifoldCF will permit to exist at one time

For our example, we're going to proceed to create a web-type repository connection definition, so go ahead and click the `Add new connection` link at the bottom of this list.

STANDARD REPOSITORY CONNECTION DEFINITION TABS

The three standard repository connection definition tabs are going to sound very familiar, because they are the same as for the output connection definition and authority connection definition: the `Name` tab, the `Type` tab, and the `Throttling` tab. We're going to fill in all of these first, before going onto the web-connector-specific tabs.

THE NAME TAB

Refer to figure 1.15 for a reminder of what the `Name` tab looks like for a repository connection definition. You set the connection definition name and description on this tab. As is no doubt beginning to sound very familiar, this name must be unique among repository connection definitions, and it is limited to 32 Unicode characters in length. The description is limited to 255 Unicode characters.

Enter a name for your web connection definition, and a reasonable description. Then, click the `Type` tab.

THE TYPE TAB

The repository connection definition `Type` tab is slightly different from other connection definition `Type` tabs. Not only does it allow you to select a connection type, but you also may select an authority group, from among those authority groups that are currently available. If you want a reminder of what this tab looks like, see figure 1.17. When you select an authority group here, it effectively specifies the governing authorities for all documents crawled using a connection made from the current repository connection definition.

The reason the authority group is selected here, and not (for example) during the creation of a job, is because it is pretty much unheard of for different documents within a repository to have different governing authorities. People just do not build content management systems that way. So, for convenience, the association is made on the repository connection definition's `Type` tab.

For our web crawling example, we can leave the authority selection alone. Select the web connector in the type pulldown, and click the `Continue` button. A large number of tabs now appear. See figure 2.17.

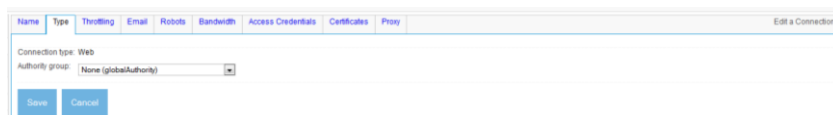


Figure 2.17 The tabs for a web connection definition. This is the most complex repository connection definition you are ever likely to see.

We'll visit many of these tabs in order to set up our web connection definition the way we'd like it. Luckily, a standard open web crawl does not involve much in the way of configuration. For more complex situations, the online documentation is a valuable resource.

THE THROTTLING TAB

The throttling tab is one of the standard tabs that every type of connection definition has. In addition to the now-familiar `Maximum connections` parameter, a repository connection definition's `Throttling` tab has something new and unfamiliar. See figure 2.18.

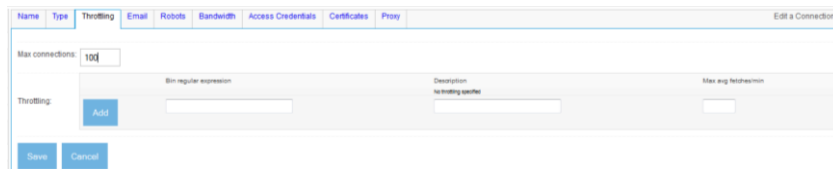


Figure 2.18 The repository connection definition `Throttling` tab. For a repository connection definition, you

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

can specify the maximum average number of fetches per minute for each and every class of throttle bin that you define.

A ManifoldCF *throttle* is just a rule. The rule applies to some set of *bin names*, described by a regular expression, and allows you to control the average number of processing operations per minute during crawling for each of the document bins that match the regular expression.

Bert's reaction to the throttling tab is perhaps predictable. "I don't understand this," he mutters. "Why can't it be easier? Why do we have to talk about bin names? Why can't we talk about servers, and things that I know about?"

You admit Bert has a point. It's not easy to understand what a bin is, since its actual definition varies from connector to connector. But you explain that since ManifoldCF deals with a wide variety of repositories, it's really not safe for it to make assumptions about how throttling occurs for each connector. So, ManifoldCF introduces an abstraction.

A ManifoldCF *document bin* is nothing more than a document category. It contains documents that all come from a single repository connection which are all related in some way. The way that these documents are related is that they are treated the same, for the purpose of throttling. Each such document bin has a bin name, which is just a string that uniquely describes that category of documents.

For example, the two web documents described by <http://foo.com/document1.html> and <http://foo.com/document2.html> are considered to be in the same document bin by the web connector, because both of these documents share the same server, `foo.com`. So the web connector gives this bin the name "foo.com", and it considers both of these web documents to be members.

Remember Any given document can be a member of more than one document bin. The rules that determine what document belongs to which document bins, and what the bins are named, are the responsibilities of the underlying connector. The goal of a document bin is to allow ManifoldCF to treat all the documents that are members of it as equivalent for the purposes of throttling.

When we fill in the regular expression part of a throttle rule, it is really a shorthand way of individually describing all the bin names we want the throttle rule to apply to. You could, for instance, specify every bin name and the corresponding maximum average fetch rate, e.g. "foo.com should have a maximum average rate of 10 documents per minute". But that would be terribly inconvenient, since there may be quite a few, and we'd need to know in advance what all the bin names were going to be, which might not be known initially. Instead, ManifoldCF gives you the ability to use regular expressions to describe whole *classes* of bin names, all at once. If a document's bin name matches a rule's regular expression, then the rule will apply. A blank regular expression will match all bin names.

Note Throttle rules are not executed in an ordered fashion. So if a bin name matches more than one regular expression, ManifoldCF uses the most conservative throttling limit from among all the matches it finds.

Regular expressions

In case you are unfamiliar with Perl-style regular expressions, which is what Java uses, they are a simple but powerful way of matching strings. For example, the regular expression `he..o` will match both the string `hello` and the string `hexyo`, and the regular expression `hi.*there` will match both `hithere` and `hi how are you there`. And there is far more you can do with regular expressions than that.

Regular expressions are very common throughout ManifoldCF, so you will definitely want to learn about them if you don't know them. There are several good web resources and tutorials out there. My favorite is <http://www.cs.tut.fi/~jkorpela/perl/regexp.html>.

Our example is going to involve crawling servers that belong to other people. Therefore, we're going to want to be sure we have proper throttling set up. Failing to do this can result in web site owners blocking access to their site from your crawling machine, because an unthrottled crawl is equivalent, in all intents and purposes, to a denial-of-service attack. You might also receive an unflattering and not very pleasant email should you overstep the bounds of crawling politeness. So what should we provide for a throttle here?

The correct answer hinges on a subtlety. There are actually two different kinds of throttling present in the web connector. The first kind, which we are looking at on the `Throttling` tab, is available for every kind of connector there is. It operates at the time that documents are handed to threads for processing, and sets an average rate for that handoff activity. The second kind of throttling we haven't seen yet, but will visit when we explore the `Bandwidth` tab. On that tab, hard limits can be set, prohibiting any access rate above a threshold that you provide.

These two different sorts of limit interact in a subtle way. If you choose to apply a limit solely on the `Bandwidth` tab, the crawl will work just fine. But what may happen is that documents will be handed to threads for processing, and the threads may be forced to just wait for some extended period of time in order to meet the throttle limits. This tends to use up threads unnecessarily. It would be more efficient, thread-wise, if the document hadn't been handed to the thread prematurely in the first place. On the other hand, you don't want the opposite situation to occur – where a document could be processed without violating the limits, but because of the random nature of the average fetch rate limit, the document doesn't get promptly handed to a thread.

Finding the right value for the `Throttling` tab can be summarized as follows.

- It has some relationship with what the maximum fetch rate is, as set on the `Bandwidth` tab;

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

- It should not be too large;
- It should not be too small.

Typically, as a rule of thumb, I set the maximum average fetch rate to be between two-thirds and nine-tenths of the maximum fetch rate. This seems to avoid the described problem scenarios.

To set this tab up correctly for our example web connection definition, first fill in a value for the `Maximum connections` field of 55. This number comes primarily from a basic characteristic of web connections; that is, there are no intrinsic connection limits involved with such connections (unlike, for example, connections to a Documentum server, which may well be limited by server configuration). For situations where there are no limits, like this one, any large number will do, but I've chosen to set the number to something not too far off of actually what was needed.

How did I do that? The answer is that I count the number of *worker threads*, which do the actual crawling work, and add a few extra connections for the others. The Quick Start example is configured by default to use 50 worker threads, so 55 should be enough to prevent the crawl from being slowed due to connection starvation. We will discuss worker threads and what they actually do in Chapter 12.

Next, we will add a throttle. Fill in a description for the throttle description, and fill in a value of 10 for the maximum average fetch rate. Leave the throttle regular expression blank – that will match every bin name. Then, click the `Add` button.

Why are we setting this value as the maximum average fetch rate for each server? The reason requires a bit of explanation. While I could say that I happen to know that the default limit on the `Bandwidth` tab is 12 documents per minute, that would not fully answer the question, because your real question is, why does the `Bandwidth` tab have such a default limit? Where did this number (and the other limits) come from?

There is no definitive reference I am aware of that tells people what **the** proper crawling limit should be. I have therefore been forced to draw on significant personal experience to set ManifoldCF's default web crawling limits. When MetaCarta used the precursor of ManifoldCF to crawl portions of the web, inevitably there would be feedback from disgruntled webmasters. While some of the feedback was of the "who are you to dare crawl us?" variety, much of it made sense, and resulted in various adjustments to the crawling parameters being used. Eventually, a cease-fire (of sorts) was reached, and I arrived at a standard set of crawling parameters. These are now encoded in ManifoldCF as the default limits for web crawling. May they serve you well.

THE EMAIL TAB

The `Email` tab is not one of the standard tabs, but is particular to web connection definitions. Its purpose is to transmit your email address to all the sites that are being crawled in a header, so that if any of these sites have a problem with what you are doing, they can notify you, and presumably convince you to desist in your undesirable activity. The

Web connector is pretty strict about making sure that you provide such an email address. However, it does not attempt to verify that the address is actually a valid one, so it is up to you to be a good citizen and provide some way of letting crawl victims potentially voice their displeasure at your activities. See figure 2.19 for a screen shot of this tab.

Figure 2.19 Screen shot of a web connection definition's email tab. Fill in your email address, not the dummy one I've entered here.

THE ROBOTS TAB

The `Robots` tab controls how the web connection works with robots.txt files. These files constitute a largely ad-hoc way for sites to tell crawlers what they are allowed to crawl and what they should stay away from. It is considered impolite for a crawler to not pay attention to robots.txt files, so the default for any web connection is to pay attention to them. See figure 2.20.

Figure 2.20 The robots tab. You can ignore robots.txt, obey it on all fetches, or only obey it for fetches of data. For the web connector, there is no difference between the latter two options.

THE BANDWIDTH TAB

The `Bandwidth` tab, as we've already discussed, allows you to place a hard limit on the fetch rate of documents. It also allows you to limit the number of socket connections, per server, as well as the actual bandwidth used for transferring documents. See figure 2.21.

Figure 2.21 The Bandwidth tab, with its default throttle already set. The default throttle parameters are usually fine for most crawls.

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

It turns out there is no need to modify these parameters for our example, since by default the connection definition is created with a reasonable throttle that was originally based on my personal experience performing crawls. But what are all these limits for? What are they designed to protect?

Simply put, limits to the fetch rate prevent a site's servers from being overwhelmed, while limits to the number of simultaneous connections prevent the crawler from using up all of the server's available socket connections. The bandwidth limitations are similarly designed to prevent the crawler from using up too much of a server's overall network capacity. The best values for these limits therefore depend on the configuration of the individual sites being crawled.

Note There are, of course, huge variations between the capacity of a small Mom-and-Pop site to absorb the overhead of a crawl, and the capacity of a large news site to do the same. You may therefore find the rather conservative default throttle limits to be too conservative for your needs. But if you are tempted to exceed them, it always makes sense to do your homework first. And, if you guess wrong, be prepared for an irate email or two.

ACCESS CREDENTIALS AND CERTIFICATES TABS

The rest of the web connection definition's tabs are used only for crawling sites that are protected in some way. Both of these tabs are highly involved in their own right, and have nothing to do with our example. We're therefore going to skip them. For a full explanation on how to set up crawls that involve certificates and credentials, I recommend consulting the online documentation, at http://manifoldcf.apache.org/releases/trunk/en_US/end-user-documentation.html.

To proceed with our example, click the `Save` button.

REPOSITORY CONNECTION DEFINITION STATUS

Any time you finish editing a repository connection definition, or if you click the `View` link from the repository connection definition list, you will see a connection definition information and status summary screen. For the connection definition we just created, it should be similar to figure 2.22.

View Repository Connection Status

Name:	Web	Description:	Web connection
Connection type:	Web	Max connections:	100
Authority group:	None (global authority)		

Throttling:	Bin regular expression	Description	Max avg fetches/min
	No throttles		

Email address:	someone@somewhere.com	Robots usage:	Obey robots.txt for all fetches
Proxy host:		Proxy port:	
Proxy authentication domain:		Proxy authentication user name:	

Bandwidth throttling:	Bin regular expression	Max connections	Max bytes/sec	Max fetches/min
	No throttles	2	64	12

Page access credentials:	URL regular expression	Credential type	Credential domain	User name
	No page access credentials			

Session-based access credentials:	URL regular expression	Login page
	No session-based access credentials	

Trust certificates:	URL regular expression	Certificate
	No trust certificates	

Connection status: Connection working

Refresh Edit Delete Clear all related history

Figure 2.22 The view page for our newly-created web connection definition. For a web connection definition, “Connection working” is always displayed.

This screen has two important purposes: first, letting you see what you actually have set up for your connection definition parameters, and second, showing you the status of a connection created with those parameters. The connection status usually represents the results of an actual interaction with the target system. It is designed to validate the connection definition parameters you’ve described. However, as is the case for the repository connection definition viewing and status screen, it can also be very helpful in diagnosing transient connection problems that might arise (e.g. invalid or expired credentials, or a crashed repository service). This makes the connection definition information and status summary screen very useful for diagnostics. For a web connection, though, the view page will always display `Connection working`, and is therefore not very useful as a debugging tool.

You may also note that there is a `Clear all related history` button on this page. As you may recall from Chapter 1, you can generate a number of history-related reports which describe what took place, when it took place, and what the result was. By default, the information needed to make up these reports is retained for 30 days, but sometimes it is helpful to remove it all, so there is a clean slate (so to speak). Click this button when you want ManifoldCF to remove all the history associated with the current repository connection.

Well, that’s all we will need as far as connection definitions are concerned. We’ve set up what we need for our example web crawl, and we’ve even taken a quick detour to see what the authority connection definition management part of the UI looks like. Next, we’ll look into the part of the UI that deals with job definition and execution.

2.3 Jobs

When you have created the connection definitions you need, the next step is to use those definitions to define the jobs you require, and run those jobs. These functions also naturally form a group within the job UI. We’ll look at job definition first, and build our example web

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

job. Then, we'll revisit the job status page to learn about all of its information and functions, and run our job.

2.3.1 Job definition

The `List all Jobs` link in the navigation area of the crawler UI brings you into the section of the UI where job definitions may be viewed, created, edited, or deleted. We are going to need to create a web job definition in order to perform our crawl.

The user interface for a job definition is fundamentally different in two ways from the user interface for a connection definition. The first way is that a job is uniquely described by an identifier, while a connection definition is uniquely described by its name. This means that more than one job of the same name may exist at the same time. The second difference is that a job's underlying connection definitions, both the output connection definition and the repository connection definition each may provide tabs to the UI. So you really can't tell what a job definition looks like without knowing what kind both of its connection definitions are. You will need to bear that in mind when you look at the screen shots of jobs in this chapter, which have been created for a job that uses the web repository connector and null output connector.

JOB DEFINITION LIST FIELDS

The job definition list screen displays all the job definitions you have, arranged alphabetically by name. Refer to figure 1.19 for a picture of what the job definition list looks like. There you can see the fields as described in table 2.9.

Table 2.9: Job list fields

Field	Meaning
Name	The name of the job definition, which is not unique
Output Connection	The name of the job definition's output connection definition
Repository Connection	The name of the job definition's repository connection definition
Schedule Type	The type of job schedule, one of "Specified time" or "Continuous crawl"

Click the `Add a new job` link at the bottom, so that we may add our web job definition.

STANDARD JOB DEFINITION TABS

When you edit or create a job definition, the job definition edit screen will contain some standard job definition tabs. These are `Name`, `Connection`, `Scheduling`, and optionally `Hop Filters`. These tabs allow you to set the job definition's description, output connection definition, repository connection definition, priority, start method, schedule, refresh interval, and many other parameters. Let's go through each of these in detail.

THE NAME TAB

You have seen what the `Name` tab looks like before, in figure 1.20. The job definition's name is really its description, and I encourage you to treat it as such. The name can contain up to 255 unicode characters, with no limitations. For our example, enter something descriptive, such as "Web job", and then click the `Connections` tab.

THE CONNECTION TAB

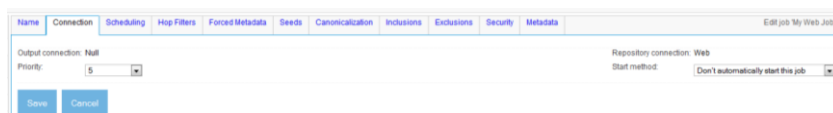
The `Connections` tab also should be familiar. See figure 1.21 for a reminder. Here you may select an output connection definition from any of those you have already created, and similarly you may choose a repository connection definition. For our example job, select the `Null` output connection definition we created in Chapter 1, and the `Web` repository connection definition we created earlier in this chapter. But wait – there are two other fields on this tab that I've not yet discussed – `Priority`, and `Start method`. What should do with these?

A job's *priority* is a number between one and nine, inclusive. The default value is five. All documents belonging to a job are considered to have the same priority value as the job has. The document's priority is meaningful only in relation to documents from other jobs. If there are documents with two different priorities in the queue at the same time, ManifoldCF will choose the documents that have the lower-numbered priority preferentially over those that have a higher-numbered priority. Effectively this means that the lower-numbered priority job is given precedence.

It is not typical at all for a user to set a job's priority to anything other than the default. The only exceptions tend to be when there is a long-running job in progress, and a user wants to run a shorter job that they need to finish very quickly.

A job's *start method* alludes to the fact that jobs in ManifoldCF may be started by hand (manually), or on a predetermined schedule (automatically). The default option is to start the job by hand. For automatic job start, you can choose from two different options. In order to select the right one, you will need to understand how job schedules work, and set up an appropriate schedule on the `Scheduling` tab. All of this is described in depth in the next section.

For our example, let's leave the job priority alone, and select the manual start option. Then, click the `Continue` button. Figure 2.23 shows what you should see.



The screenshot shows a web form with several tabs: Name, Connection, Scheduling, Hop Filters, Forced Metadata, Seeds, Canonicalization, Inclusions, Exclusions, Security, and Metadata. The 'Connection' tab is active. It contains two dropdown menus: 'Output connection: Null' and 'Repository connection: Web'. Below the first dropdown is a 'Priority' field with a value of 5 and a small arrow icon. Below the second dropdown is a 'Start method' field with the value 'Don't automatically start this job' and a small arrow icon. At the bottom left are 'Save' and 'Cancel' buttons. At the top right is a link 'Edit job: My Web Job'.

Figure 2.23 The tabs for a web job. Believe it or not, they all do something essential.

Finally, click on the `Scheduling` tab.

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

THE SCHEDULING TAB

The scheduling tab is quite important, and is one that we have not yet had the opportunity to consider in detail. For a screen shot, see figure 2.24.

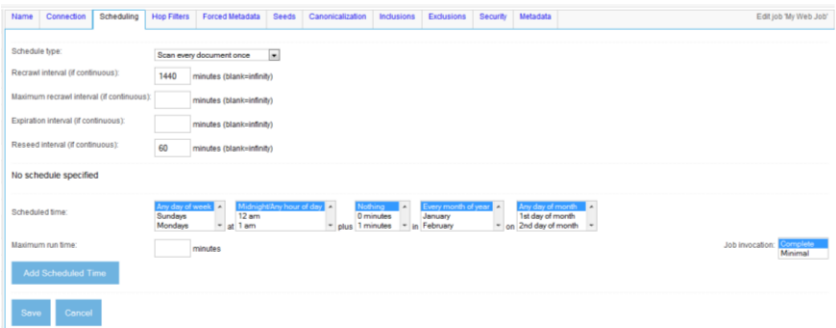


Figure 2.24 The Scheduling tab. This tab has two sections. The top section allows you to set the schedule type and parameters, and the bottom controls when the job runs.

The fields on this tab belong to two distinct categories. The first category, represented by the top part of the tab, controls what crawling model the job uses, and associated parameters. Specifically, you can select here either a standard incremental crawl, which will run through to completion every time the job is run, or a continuous crawl, which will run forever (or until it is aborted). If you select the continuous crawling option, you also can supply document scheduling parameters for that crawl, such as how often to rescan a document, when to expire it, and how often to accept new seeds into the crawl. Table 2.10 describes these parameters.

Table 2.10 Document scheduling related fields and their functions

Field	Function
Schedule type	Select either Scan every document once or Rescan documents continuously.
Expiration interval	This is the time, in minutes, from a document's <i>origination time</i> until the time it should be removed from the system. A document's origination time is up to the underlying connector to determine, but defaults to the first time the document was fetched. If a document remains referenced by another document after it was expired, and its origination time is based on fetch time, then the document may be fetched again at that point. This parameter is only meaningful for continuous crawling.
Recrawl interval	This is the minimum amount of time, in minutes, between fetches of a document. The actual interval may be longer based on the history of changes for the

	document. This parameter is only meaningful for continuous crawling.
Reseed interval	The reseed interval is the amount of time, in minutes, between updates of a job's queue based on seeds specified in the job. The seeds used by the job are a function of what connector is in use, and may or may not be directly under user control. This parameter is only meaningful for continuous crawling.

The second section of the form controls when the job itself executes. Here, you may create a number of *schedule records*, each of which consists of a combination of dates, times, and days of the week, which taken together describe zero or more points in time. The pulldowns for the time and date fields are all multivalued, so you can select more than one of each. For example, in one record you could select Saturdays and Sundays at 1 AM, or the 1st and 15th of the month at midnight. (Of course, you might instead pick a combination of values from the pull-downs which could never occur in real life, such as February 30th. We will discuss what happens if you do something like that later in this section.) You may also add more than one schedule record, if your scheduling needs are too complex to be described by a single such record.

Each schedule record also allows you to specify a time interval, which is labeled *Maximum run time* on the form. If you fill in this value, then, for each point in time that matches the time and date values you selected, a time window from that point extending the number of minutes you specified is defined. If the job is defined as being started automatically (see the *Connection* tab), it is allowed to run only within a valid time window, and it will pause itself, by entering a special state called *waiting*, at the end of the window. The job will then resume at the beginning of the next valid time window that occurs.

For example, suppose you wanted to allow your job to run from 1 AM to 3 AM on Saturday and Sunday nights. You can readily specify the start times using one schedule record by selecting the *Saturday* and *Sunday* values in the day-of-the-week pulldown, and the 1 AM value in the hour-of-the-day pulldown. (You could also select the 0 value for the minutes-of-hour pulldown, but that is already assumed to be the default.) Then, all you need to do to complete the schedule record is to fill in a value of 120 for the *Maximum run time* field, and click the Add button.

An astute reader may notice that it is possible for a person to create a set of schedule records where the intervals abut or overlap each other. What does ManifoldCF do in that case? The answer depends on the setting you chose for the *Start method* pulldown on the *Connection* tab. There, in addition to whether you wanted to start the job automatically or not, you could choose whether to start a job only at the beginning of a window, or start it anytime within a window. If the former option was chosen, then the only time a job may be automatically started is at a date and time that explicitly matches what is specified for all the schedule records for the job. If the latter option is chosen, then as long as the job is not already running, and the time and date is within any valid window, the job will start up. So

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

ManifoldCF really does not care if your time windows overlap; it really mostly cares about when the windows begin.

For our example, we intend to start the job manually, so we will not need to define a schedule. But we could go either way on the question of whether to run our job continuously or not. In a real world situation, this would depend in large part on the actual web crawling task we choose to undertake – whether it is going to be a task of a well-known and limited size, or whether it could potentially be a very large and open-ended task. Since one of the hallmarks of web crawling is the latter, let's set things up as if that is what we will be doing. Select the value `Rescan documents dynamically` from the `Schedule` type pulldown.

We will also need to select parameters appropriate for this kind of crawling for the rescan interval and the expiration interval. For continuous crawling, there are really two different models one could choose to use. The first model is to refetch each document multiple times, at some interval between fetches, in order to look for changes or deletions. The second model is to fetch each document just once, and then get rid of it when it is older than some specific period of time. The choice of model once again depends on our particular task; if we are mostly interested in new content, for instance, then the expiration model works pretty well, while if what we care about is older content, then perhaps the refetch model is a better fit. For our example, it is older content we will be crawling, so we will choose the refetch model. The default values for all of the interval parameters are set up for exactly this situation – a recrawl interval of 1440 minutes (1 day), and an infinite expiration interval. Let's accept the defaults, and proceed by clicking the `Hop Filters` tab.

THE HOP FILTERS TAB

For some kinds of crawls, the only way to control the number of documents involved in the crawl is to put a limit on the number of hops that must be taken in order to reach the document from a starting point, or seed, document. This is a standard feature of most crawl-once-model web crawlers.

The good news is that ManifoldCF also supports this functionality. The bad news, however, is that you may not actually want to make use of it. The reason for this caveat is that incremental crawling makes hopcount-based limits difficult to implement properly, and even though ManifoldCF offers a "correct" implementation consistent with the incremental model, the bookkeeping expense required is significant.

Imagine, for example, what might happen if you are crawling the web, and the documents you are going after are all interlinked so that there are two different paths to a document. For convenience, let's pretend there is a single seed called S. You can get to document D either by the path S -> A -> D, or by the path S -> B -> C -> D. See figure 2.25.

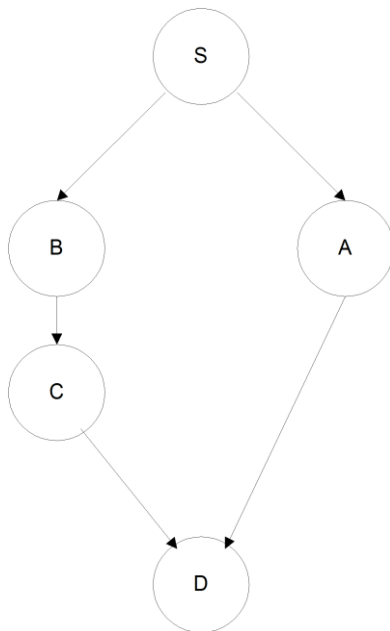


Figure 2.25 Hypothetical relationships between web documents S, A, B, C, and D. There are two ways of getting to document D that have different hop counts.

The first problem occurs if, by chance, the crawler fetches document C before it fetches document A. Then, it may make the decision to exclude document D, based on the number of hops from S to D by way of C. Later on, when document A is fetched, it will need to revisit that decision, and may then change its mind and decide to fetch D after all.

Consider now what happens when the crawler has already fetched D, and then, either due to continuous rescanning, or due to another run of the job, the crawler determines that A has vanished. Clearly, the correct thing to do will be to reconsider whether or not to keep D, or remove that from the crawl as well. The problem becomes even more difficult when both A and B go away, because neither C nor D are reachable by way of the seed at all. Strictly, then, they should both be removed, if hop count is a document filtering consideration.

But keeping enough information around to allow ManifoldCF to handle deletions like this properly is not easy. It requires approximately ten times the number of database records to handle this case than would otherwise be needed for crawling the same job with the same documents, because all the links between documents must be tracked, and the document dependencies on other documents must be recorded.

If you click on the `Hop Filters` tab, you will see something similar to figure 2.26.

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

Figure 2.26 The Hop Filters tab. This tab lets you specify the maximum hop count from a seed document for links of various types. The types are determined by the underlying connector. If the connector supports no link types, the Hop Filters tab will not appear.

The upper part of the tab is where you would set specific limits for the number of hops from a seed, counting hops of the specified kind. It is up to the connector to provide a list of what types of hops it might support for this purpose. If you leave the value blank, then ManifoldCF will enforce no limit on the allowable hops of that kind from seed to document.

What connectors support filtering by hop count?

As of this writing, only two of the ManifoldCF repository connectors support any kind of hop counting: the web connector, which supports two kinds of hops – “link” and “redirect”, and the file system connector, which supports one – “child”. The feature was enabled solely for demonstration in the file system connector; the web connector is the only one that has any real use case for document filtering by hop count.

When you think about it, this is not surprising. The web repository is actually a very specialized kind of repository, because you have no way of knowing in advance what the graph being crawled will look like. The relationships between documents are manifold, and the number of documents you could reach from a single seed with an unlimited number of hops is very large. No other repository that I am aware of has this kind of scale or structure.

The bottom part of the tab is where you can choose to disable various aspects of hop counting, usually for reasons of performance. The default selection, `Delete unreachable documents`, tells ManifoldCF to keep a consistent, accurate hop count at all times. Documents whose hop count suddenly increases as a result of changes will be deleted if the hop count filtering criteria is violated. As we have already discussed, this is a quite expensive thing to do.

The second option, `Keep unreachable documents for now`, will allow your crawl to be considerably more efficient. When you select this option, the bookkeeping necessary for getting rid of unreachable documents is still performed, but no actual cleanup of such documents is undertaken. However, with this option, you could change your mind at a later point, and ManifoldCF would then go ahead and clean out unreachable documents as it discovered them.

The last option, `Keep unreachable documents forever`, is the one I recommend that you use for web crawling, if you do not need to set any hop filters. It is by far the most efficient, by virtue of the fact that it doesn't attempt to keep track of any of the bookkeeping needed to detect disconnected parts of the crawl graph. The downside of this option is that, once you have accumulated documents that are no longer reachable from the crawl seeds, there is no way for ManifoldCF to go back and remove them, because the information is not kept to do that. This is, however, what many other crawlers do, so for most users it is probably an acceptable way to go.

With the `Hop Filters` tab, you have now concluded your exploration of a job's standard tabs. Click the `Seeds` tab next to start setting up the details of our crawl.

THE FORCED METADATA TAB

The `Forced Metadata` tab is shared by all connection types. Its purpose is to allow you to send along, with every document, some amount of job-specific information to the output connection. This can be very helpful if, for example, you have a highly configurable search engine, such as Apache Solr, for the back end index, and you want to process documents from one particular source differently from other sources. The `Forced Metadata` tab gives you a way to hint about the source of the data to the back end. See figure 2.27.

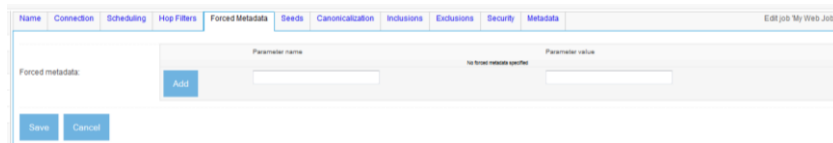


Figure 2.27 Add whatever fixed metadata you want, one value at a time. Don't forget to click the “Add” button.

The `Forced Metadata` tab is a relatively new addition to the core functionality of the ManifoldCF framework. You should therefore not be surprised to find other tabs associated with specific connection types that allow you to do essentially the same thing. These other tabs are kept for backwards-compatibility reasons. We'll get to that in a little bit.

THE SEEDS TAB

The `Seeds` tab is specific to the web connector. This is where you provide the starting URLs, or seeds, that the crawler will use to spider to every other document in the set. See figure 2.28 for an idea as to what this tab looks like.

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

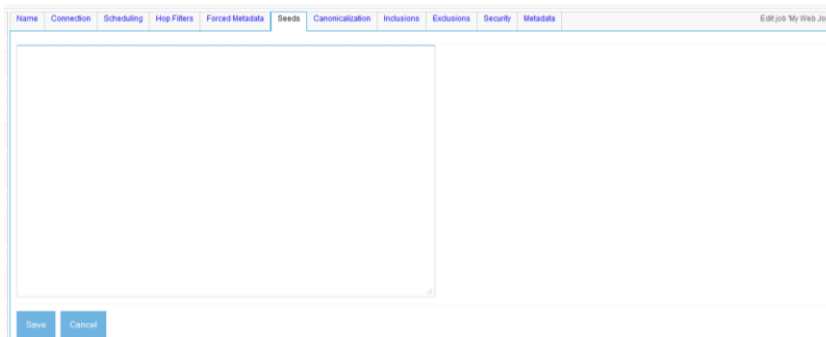


Figure 2.28 The seeds tab. The text area is a blank place where you can type in your seed urls, separated by line breaks. You can also include comment lines, which start with the “#” character. The UI is designed to make it easy to cut and paste.

For our example, what seeds should we choose? You can, of course, supply your own, but if you want some safe ideas, you can always crawl some of the Apache sites. You can even start with the ManifoldCF site, at <http://manifoldcf.apache.org>, and the Solr site, at <http://lucene.apache.org/solr>.

Enter these two URLs into the Seeds field, and then click the `Canonicalization` tab.

THE CANONICALIZATION TAB

The `Canonicalization` tab allows you to control how the web connector canonicalizes URLs. *Canonicalization* is the process of converting URLs that are different but which describe the same document into a single URL in a standard form. This is important because without some notion of URL mapping, the web connector may wind up fetching the same document many times. This is often a problem with dynamically generated sites.

The tab itself allows you to specify a set of rules, which are assessed in order. Each rule has a regular expression which determines if the rule should apply to a given URL or not. If the rule applies, you may choose how the URL will be normalized, such as whether arguments may be reordered, or URL-based cookies of various forms may be removed. For example, if you want to crawl a specific site that is dynamically generated using a Java application server, you may want to add a rule just for that site, and select canonicalization that includes argument reordering and Java application server cookie removal.

See figure 2.29 for a screen shot of the `Canonicalization` tab.

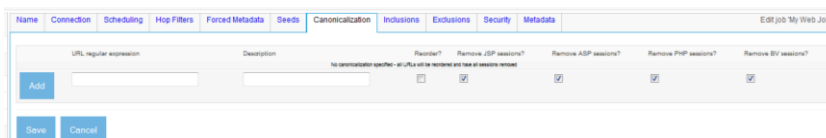


Figure 2.29 The Canonicalization tab of a web job. Use this tab to control the canonicalization of URLs.

To use this tab, fill in a URL regular expression corresponding to the site whose canonicalization you want to control. You may add a description if you want to remember later what you were trying to do. Then, select the checkboxes that you want to apply for the rule, and click the **Add** button to add the rule to the rule list.

Since the sites we are crawling are statically generated, we have no reason to change anything, so let's just continue on. Click the **Inclusions** tab.

THE INCLUSIONS TAB

The **Inclusions** tab is the place in a web job where you can control what documents are going to be considered part of the job, based on their URLs. The tab itself is relatively simple. See figure 2.30.

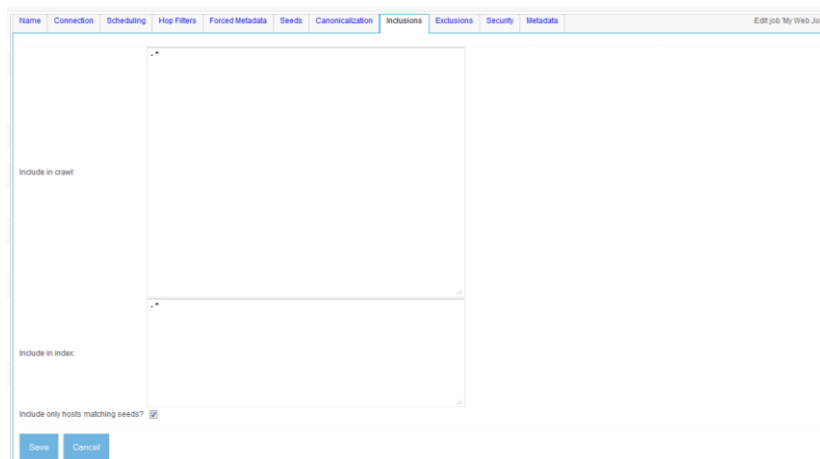
The screenshot shows a web application interface for configuring a web job. At the top, there is a horizontal menu with tabs: Name, Connection, Scheduling, Hop Filters, Forced Metadata, Seeds, Canonicalization, Inclusions (which is currently selected), Exclusions, Security, and Metadata. Below the menu, the main area is divided into two sections. The first section is labeled 'Include in crawl:' and contains a large text input field with a placeholder '.*'. The second section is labeled 'Include in index:' and also contains a large text input field with a placeholder '.*'. Below these two sections, there is a checkbox labeled 'Include only hosts matching seeds?' which is currently checked. At the bottom left of the form, there are two buttons: 'Save' and 'Cancel'. In the top right corner of the interface, there is a link that says 'Edit job "My Web Job"'

Figure 2.30 The Inclusions tab of a web job. Fill in newline-separated regular expressions, and/or click the checkbox to limit the URLs to those that share the same host names as the seeds.

To use this tab, you provide some list of regular expressions, all separated by newlines, which together specify the set of URLs that you want to crawl. You can separately specify the URLs that you crawled that you also want to index. You may also choose to limit the URLs crawled to those matching the hostnames of the seeds you provided in the **Seeds** tab by checking the **Include only hosts matching seeds** checkbox.

For our example, we don't want to do a broad web crawl, so click the checkbox, and leave the other two fields with their default values. Then, click on the **Exclusions** tab.

THE EXCLUSIONS TAB

The purpose of a web job's **Exclusions** tab is to allow you a way to filter out URLs that make no sense for indexing. URLs are already filtered to some extent by way of their content type; the output connection definition you specified provides a list of valid content

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

types to any repository connector that can make use of it. However, documents still must be fetched in order to find out their content types, so it always a good idea to provide a basic level of URL filtering upstream of that.

See figure 2.31 for a screen shot of what the `Exclusions` tab looks like. No surprises here; it looks very much like the `Inclusions` tab.

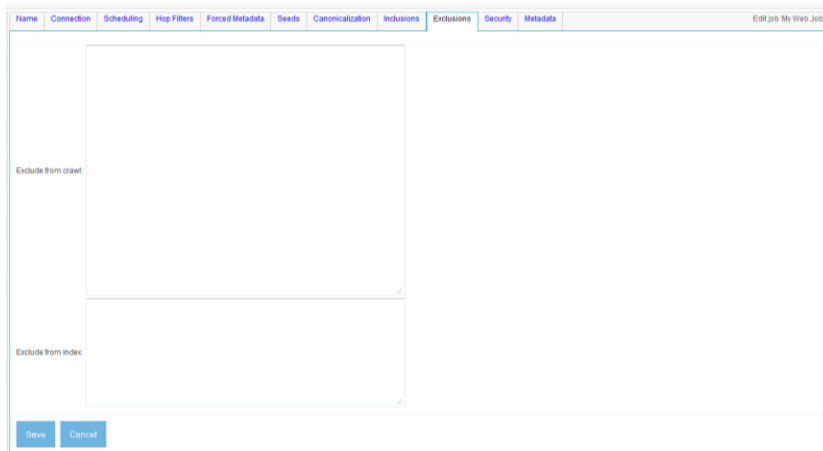


Figure 2.31 A web job's `Exclusions` tab. Enter here newline-separated regular expressions describing the URLs to be excluded from the job.

Once again, you can separately specify the URLs to exclude from the crawl, and those to exclude from the index. For our example job, we will want to exclude most image forms. In general, that will include all URLs that end with `.png`, `.jpg`, or `.gif`. Let's go ahead and write appropriate regular expressions for these, and enter them:

```
\.gif$  
\.GIF$  
\.jpg$  
\.JPG$  
\.png$  
\.PNG$
```

THE SECURITY TAB

A web crawl does not usually involve securing documents, but in intranet situations, there are sometimes instances when it would be useful to protect documents on a job-by-job basis. ManifoldCF lets you do this for most connectors, with some work.

The way security works in ManifoldCF is by the use of *access tokens*. Access tokens are attached to a document when it is indexed. They are also returned by the ManifoldCF authority service, which then allows the search index to exclude results that a user should not see. We'll talk more about this in Chapter 4.

For connection types such as the Web Connector, you can provide your own access tokens for every document described by the job. For this connection type, this is done on the **Security** tab. See figure 2.32.

Figure 2.32 You can add individual access tokens to a document, if you know what they should look like. You need to know a lot about the specified authority group in order to do this, however.

This functionality is really “expert”, in the sense that you need to know what an access token for your particular authority(ies) will look like before you can hope to add anything appropriate. Normally this is something a ManifoldCF user does not need to worry about at all.

THE METADATA TAB

Remember when I was mentioning how some tabs were kept around for backwards-compatibility reasons? Well, that is certainly true for at least a portion of the web connector type’s **Metadata** tab.

On this tab, you can do two things. The first is to do exactly the same thing as what the **Forced Metadata** tab lets you do. This functionality is maintained simply to not break backwards compatibility with earlier versions of ManifoldCF. The second thing you can do, however, is to exclude individual headers from metadata. Normally, the web connector type includes all headers, except for a select few that have nothing to do with the web document’s content, as document metadata. But there are some headers that can go either way. Some sites consider these headers to be part of the data, while others consider them to be part of the fetching process. See figure 2.33.

Figure 2.33 Another way to add forced metadata. But you can also exclude specific headers from the document metadata.

With the metadata exclusion list, we’re finally done creating the job! Click the **Save** button. You should see the view page for the job, similar to figure 2.34.

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

Name:	My Web Job	
Output connection:	Null	Repository connection: Web
Priority:	5	Start method: Don't automatically start
Schedule type:	Scan every document once	Maximum recrawl interval: Not applicable
Minimum recrawl interval:	Not applicable	Reseed interval: Not applicable
Expiration interval:	Not applicable	
No scheduled run times		
No forced metadata		
Maximum hop count for link type 'link':	Unlimited	
Maximum hop count for link type 'redirect':	Unlimited	
Hop count mode:	Delete unreachable documents	
Seeds:	http://lucene.apache.org/solr http://manifold.apache.org	
No canonicalization specified - all URLs will be reordered and have all sessions removed		
Include only hosts matching seeds? yes		
Include in crawl:	.*	
Include in index:	.*	
Exclude from crawl:	\.gif\$ \.css\$ \.png\$ \.mp3\$ \.png\$ \.png\$	
Exclude from index:		
No access tokens specified		
Excluded headers:		
No metadata specified		
<input type="button" value="Edit"/> <input type="button" value="Delete"/> <input type="button" value="Copy"/> <input type="button" value="Reset seeding"/>		

Figure 2.34 Our example web job, with all the specification information we've set. We're ready to give it a try.

This looks good! It appears to be a complete and correct summary of all the decisions we've made during the job definition process.

But, what is the meaning of the `Reset Seeding` button at the bottom? Well, as we've discussed, ManifoldCF is an incremental crawler. That means that it must keep track of what it has done for a given job in past job runs. One of the ways it keeps track of this is to remember the time that it last checked for new documents to crawl. In that way, for many repositories ManifoldCF is able to only consider documents that have changed since the last job run. But, in some cases, such as when (for instance) properties of the target index are changed, it is helpful to make ManifoldCF repeat itself. The `Reset Seeding` button does exactly that. It is meant to be used primarily during the development process.

We're ready to move on to look at the job status page, and start our job.

2.3.2 Job status and execution

Proceeding downward in the UI's navigation area, the next link we encounter is the `Status` and `Job Management` link. This link takes you to a page you have already seen once: the job status page. See figure 1.26 to refresh your memory as to what this page looks like.

In chapter 1, we explored this page only so far as was needed to run a single job to completion. There are, however, many details that I deliberately overlooked at the time to prevent undue confusion. Now we will go back and learn the details.

STATUS LIST FIELDS

We have already summarized the status fields in table 1.1. But we still need to explain what all the job statuses mean. Table 2.11 should clear things up.

Table 2.11 Job statuses and their meanings

Job status	Meaning
Not yet run	The job has never been run before.
Starting up	The job has not started running yet, but is making preparations to run.
Running	The job is running.
Running, no connector	The job is running, but the underlying connector has been uninstalled, so nothing is actually happening.
Aborting	The job was manually aborted, or encountered an error that required it to be stopped, but has not yet finished aborting.
Restarting	The Restart link for the job was clicked, and the job is in the process of restarting itself.
Done	The job has either completed successfully, or was aborted and is now inactive.
Stopping	The Pause link for the job was clicked, but the job has not yet paused.
Paused	The Pause link for the job was clicked, and the job is paused.
Resuming	The Resume link for the job was clicked, but the job has not yet resumed.
Waiting	The job is running on a schedule, and the current time is outside all of the job's schedule windows.
Terminating	The crawl completed, and the job is removing old documents before it finishes.
End notification	The job is either finished, or has finished aborting, and the output connector is being notified of this completion. (The Solr output connector uses this opportunity to commit documents to its index, for example.)
Cleaning up	The job was deleted, and all documents associated with the job are in the process of being deleted from the output index.

Some of these statuses are the result of a user intervening in the job's progress by means of the crawler UI. The leftmost, unlabeled column in figure 1.26 shows an area where links appear that a user may click to intervene in a job's progress. Only certain links are

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

displayed at any given time, based on the job's status. But let me introduce them all to you, one at a time.

THE START LINK

We've already seen the `Start` link in action in Chapter 1. This link appears only when the job is not running, and is not paused, waiting, or being deleted. Clicking the link will begin the process of starting the job. This action is sometimes given the rubric "starting a job manually".

A job that is run to completion using this link will fully synchronize the repository and the target output index. All documents that need to be added or updated or deleted will be added, updated, or deleted, regardless of the characteristics and limitations of the underlying repository. To do that, as we have discussed in Chapter 1, ManifoldCF may need to check every document for changes or deletions. But that can be a time-consuming operation, often too time-consuming for some real-world freshness constraints. Luckily, ManifoldCF has a solution to that problem too.

THE START MINIMAL LINK

The `Start Minimal` link appears whenever the `Start` link appears. Clicking this link, however, does not guarantee full synchronization of the repository and the target output. Instead, the function of this link depends on characteristics of the underlying repository connection type – specifically, the connector model, which we'll talk about further in Chapter 7. Only those documents which the repository itself has flagged as having been added, changed, or deleted will be processed. Since most repositories do not keep track of deleted documents, often the effect of this link is to crawl only those documents that have been added or changed. Deletions will be postponed until the next time a full crawl is done, via the `Start` link or by the equivalent schedule.

We're going to need to start our example web job manually, because we did not set it up to start on a schedule. Click the `Start Minimal` link next to the job you just created, and it should start. When the job status page refreshes, you probably will see that the job appears first in the `Starting up` state. See figure 2.35.

Status of Jobs

[Restart](#) [Restart minimal](#) [Abort](#)

Name	Status	Start Time	End Time	Documents	Active	Processed
My Web Job	Starting up	Not started	Never run	0	0	0

[Refresh](#)

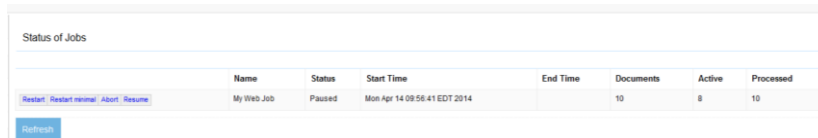
Figure 2.35 Starting the web job. Note the job status of "Starting up".

THE PAUSE LINK

The `Pause` link appears only when the job is running. Pausing the job causes all of the threads that are working on the job's documents to stop what they are doing, and accept no further work for that job until its state is changed. You might use this link if you wanted to

debug another job, and didn't want any potential interference or confusion from an already-running job.

You can go ahead and click the example job's `Pause` link, as soon as it appears. You may need to refresh the job status screen a few times before the job is truly in the `Running` state and hence can be paused. When you do, the page should look like figure 2.36.



The screenshot shows a web interface titled "Status of Jobs". It contains a table with columns: Name, Status, Start Time, End Time, Documents, Active, and Processed. A single row is visible for "My Web Job" with status "Paused" and start time "Mon Apr 14 09:56:41 EDT 2014". To the left of the table, there are links: "Restart", "Restart+resume", "Abort", and "Resume". The "Resume" link is highlighted in blue. Below the table, there is a "Startpage" button.

	Name	Status	Start Time	End Time	Documents	Active	Processed
Restart Restart+resume Abort Resume	My Web Job	Paused	Mon Apr 14 09:56:41 EDT 2014		10	8	10

[Startpage](#)

Figure 2.36 A paused web job. The job status screen now displays the `Resume` link.

THE RESUME LINK

Clearly, after a job is paused, there must be a way to resume it. The `Resume` link appears only on jobs that have been paused. Clicking it will resume the job.

Go ahead and click the `Resume` link for the example web job. The status should immediately return to `Running`, and the document counts should continue to climb.

THE ABORT LINK

When a job is running, and is not aborting, and is not in the process of finishing up or being deleted, an `Abort` link appears. Clicking this link will begin the process of aborting the job.

Aborting a job may take some time, because not only do all the threads that are working on the job's documents need to stop what they are doing, but there is other bookkeeping which needs to be done. This can take some time, so you may need to be patient.

We are going to eventually use the `Abort` link to stop our continuous example job, since there is no other way to stop a continuous job, but we will do this only after it has downloaded and indexed a sizeable amount of content, and after we've looked at a number of status and history reports.

THE RESTART LINK

Whenever you see the `Abort` link, you will also see a `Restart` link. Clicking the `Restart` link is the equivalent of clicking `Abort`, waiting for the abort to finish, and then clicking `Start` again.

Why would you ever want to do this? The usual case is when you've made some changes to the job's document specification while it is already running, and you already know you are going to need to run the job again when it is done. Restarting the job means you can do it all in one click.

THE RESTART MINIMAL LINK

Whenever you see a `Restart` link, you will also see a `Restart Minimal` link. Clicking it is the equivalent of clicking `Abort`, waiting for the abort to finish, and then clicking `Start Minimal`.

That's it for the job status page. Next, we'll look at the two kinds of reports that we can use to understand what our job is doing, or what it has already done. These help us to answer the question, "what will happen next?"

2.4 Status reports

There are two status reports, grouped together, on the crawler UI navigation area. A status report is basically a view into the current *job queue*. The job queue is the set of records that the crawler uses to keep track of what documents it is crawling, or has crawled in the past. It is the principle data structure that determines what the crawler's plans are. Figure 2.37 shows pictorially how data flows through ManifoldCF during a crawl, and generally how the job queue is populated and maintained.

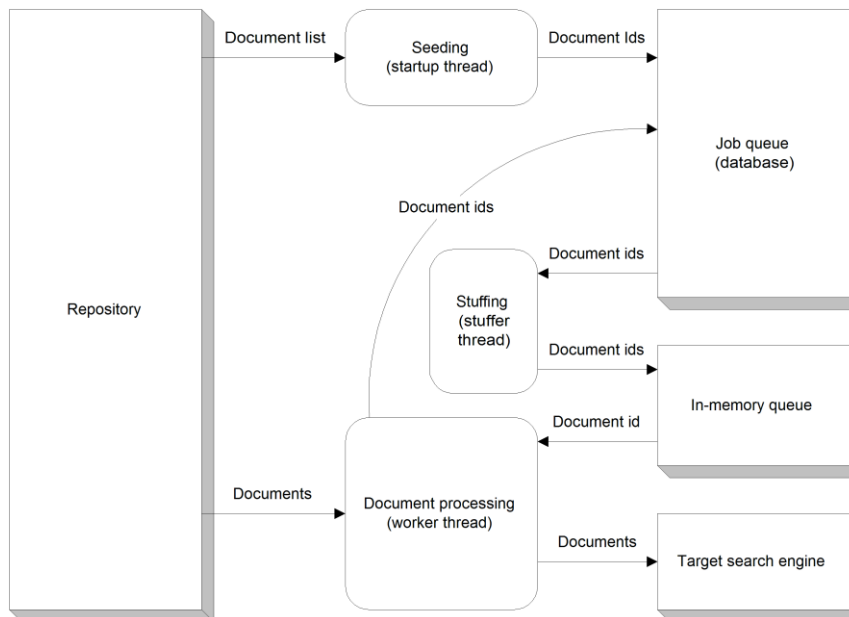


Figure 2.37 The flow of data through ManifoldCF. Note the role of the job queue in tracking documents via their identifiers.

Note This diagram is intended mainly to help you put what you are learning about the UI into some kind of overall context. There are data structures and activities presented which we've only barely mentioned, such as document processing, or have not mentioned

at all yet, such as document stuffing. We will be covering all of this in more detail in Chapter 11.

Since all status reports work against the job queue, which has a fixed structure, all status report UI pages have a similar appearance. The majority of each report page is used to select which queue records to consider for analysis. Only very limited additional information is needed that is specific to the actual kind of report it is.

2.4.1 Document status report

The document status report is just a simple listing of the matching records in the job queue. Figure 2.38 is a screen shot of an initial document status report screen.

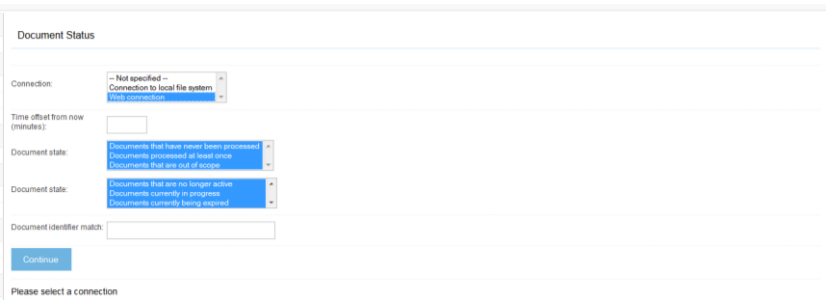


Figure 2.38 A document status report, before the connection definition has been selected. To select which jobs you want to examine, you must select the connection definition and click the Continue button.

Here you are expected to select your desired connection definition and click the Continue button. The other fields are set up by default to include all queue records. You may alter them, of course, to more exactly specify what you are seeking in the report. Table 2.12 lists these fields and describes what they do.

Table 2.12 Status report fields for job queue record selection

Field	Description
Connection	This is the repository connection definition you are interested in looking at.
Jobs	Select the specific jobs associated with that repository connection definition whose job queue records you want to see.
Time offset from now	Enter a time offset from now, in minutes, either positive or negative, that will be used to determine the document status value for each individual record during the

Chapter author name: Wright

	record selection process. Leave this blank for no time offset.
Document state	Select either documents that have never been processed, or were processed at least once, or both.
Document status	Select the status of the document, in conjunction with the current time and the Time offset from now field. Status values are described further in table 2.12.
Document identifier match	Enter a regular expression which must match the document identifier for a record in order for the record to be included. A blank regular expression will match all identifiers.

The meaning of the document statuses you can select from is summarized in table 2.13.

Table 2.13 Document status values and their meanings

Status value	Meaning
Documents that are no longer active	This status covers documents that have already been processed in the current job run, that will not need to be processed again.
Documents currently in progress	This status includes documents that have been begun being processed, but whose processing is not yet completed.
Documents currently being expired	This status includes documents that have been handed to expire threads for expiration but whose expiration has not yet been completed.
Documents currently being deleted	This status covers documents that have been handed to document removal threads for deletion, but whose deletion has not yet been completed.
Documents currently available for processing	Documents with this status are ready to be processed, based on their assigned priority.
Documents currently available for expiration	Documents with this status are ready to be expired, but have not yet been handed to an expire thread.
Documents not yet processable	Documents with this status have been placed in the queue with a time value before which they cannot be processed, and it is not yet that time.
Documents not yet expirable	Documents with this status have been placed in the queue with a time value before which they cannot be expired, and it is not yet that time.

Documents waiting forever	Documents with this status have been placed in the queue with no ability to be processed or expired, and are thus waiting forever. Documents in this state usually indicate a bug in a connector.
Hopcount exceeded	Documents with this status are not indexed, but remain in the queue as markers, in order to keep track of their hop counts.

Once you have selected your connection definition and clicked the `Continue` button, you should be able to select one or more jobs. Once you do that, and click the `Go` button, you will see a screen similar to figure 2.39.

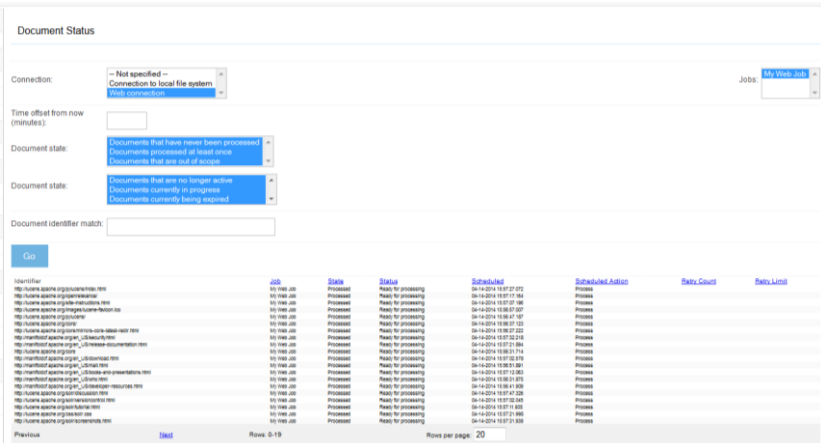


Figure 2.39 The document status for an in-progress web crawl. The “Scheduled” column is displaying a value of zero milliseconds since Jan 1, 1970, 12:00 AM GMT for most of these queue records, but is being converted to the local time zone.

The fields you see in this screen shot are described further in table 2.14.

Table 2.14 Document status report columns and their meanings

Column	Meaning
Identifier	This is the document identifier.
Job	This is the name of the job that the document belongs to.
State	This is the state of the document, which is either never been processed, or processed at least once.
Status	This field will list one of the values described in table 2.13.

Chapter author name: Wright

Scheduled	This field contains a timestamp. The document cannot be acted upon until the current time is at least greater than the timestamp.
Scheduled Action	This field tells you what ManifoldCF will try to do with the document next. This will be either <code>process</code> or <code>expire</code> .
Retry Count	This field lists the number of remaining times the document may be retried, before being considered unprocessable.
Retry Limit	This field lists the cutoff time after which a failed processing attempt will make the document be considered unprocessable.

In this report, you may also notice that some of the column headers are clickable links. Clicking on a column heading will cause ManifoldCF to sort the report by that field. You may also change the number of records displayed by modifying the value in the `Rows per page` field, and clicking the `Go` button. Or you may choose to page through the displayed records, if there are enough of them, by using the `Previous` and `Next` links.

2.4.2 Queue status report

The queue status report also operates on the job queue, and has a great deal in common with the document status report. All the record selection fields and processes are identical. But what the queue status report does with the selected records is to classify them into groups, based on their document identifiers and a regular expression that you provide, and count the number of documents of each status value in each group.

Figure 2.40 presents a screen shot of the queue status report for our example web job, with the default queue record selection criteria, along with an `Identifier class description` field set to `()`.

Figure 2.40 A queue status report for a web job, with an “Identifier class description” value of `()`. All the individual records are added up in this case.

The `Identifier` class description field could use some explanation. This field contains a regular expression that allows you to identify the individual groups that will be displayed, by means of parentheses. For example, suppose that you are doing a web crawl, and your document identifiers are therefore URLs. You want to display the queue status summary for each server. So, your identifier class description regular expression must somehow extract the server portion of each URL in order to establish the desired classification of the URL. We accomplish this by using parentheses around the match area of interest in order to bring it to ManifoldCF's attention. This is a regular expression feature commonly known as *grouping*.

Looking at an example url, <http://foo.com/something>, it seems clear that we can write such a regular expression easily enough. We want to exactly match the protocol portion of the url, and include what comes after it up to the first `/`, `?`, or end-of-string value in our group. The corresponding regular expression looks something like this:

```
^https?:\/\/([^\/?$]*)
```

Go ahead and change the `Identifier` class description value to the one above and click the `Go` button. What do you see? Is it what you expected?

Note It's always a good idea to try out your regular expressions to be sure they do exactly what you think they will. I have found that this site, <http://www.cis.upenn.edu/~matuszek/General/RegexTester/regex-tester.html>, is particularly useful in that regard. It also supports and demonstrates grouping well.

The columns displayed in the queue status report are described in table 2.15.

Table 2.15 Queue status report columns and their meanings.

Column	Meaning
<code>Identifier Class</code>	This is the group string, as described by the identifier class description.
<code>Inactive</code>	This is the count of documents in the group that are in the "inactive" state.
<code>Processing</code>	This is the count of documents in the group that are in the "processing" state.
<code>Expiring</code>	Displays the count of documents in the group that are in the "expiring" state.
<code>Deleting</code>	This is the count of documents in the group that are in the "deleting" state.
<code>About to process</code>	Counts the number of documents in the group that are in the "about to process" state.
<code>About to expire</code>	This is the number of documents in the group that are in the "about to expire" state.

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

Waiting for processing	Displays the number of document in the group that are in the “waiting for processing” state.
Waiting for expiration	Counts the number of documents in the group that are in the “waiting for expiration” state.
Waiting forever	Counts the number of documents in the group that are waiting forever.
Hopcount exceeded	This is the count of documents which have the “hopcount exceeded” status.

Status reports clearly present current information from the job queue. But, what if we want information about what happened in the past? A status report is not going to help us remember events that occurred before the present. For that, we need a history report.

2.5 History reports

A history report looks at records kept of events from the past. These events have taken place while your jobs were running, and they have been recorded in the database for the sole purpose of understanding what has happened. In ManifoldCF, such historical events are called *activities*, and they include generic events like job start, job end, document indexing, document index deletion, in addition to specific activities that are defined by individual repository or output connectors. For example, if you are trying to diagnose a problem with how your web connection definition’s session authentication is set up, it’s most likely that you will need to see what happened, so you will want to look at a history report.

All history reports have a similar overall structure. Most of what you do in the UI is to specify what history records to include in the report’s analysis. Since the data in each record is the same for all of the reports, this means that the report pages are quite similar to each other. The differences are relatively small.

Please review figures 1.28 and 1.29 for a reminder of the kinds of information you may use to select records when setting up a history report. You start by selecting the connection definition, because all history records are attached to a repository connection definition. But you may also select a start time, an end time, and a subset of the possible activities to include in the report. Finally, you can provide a regular expression to match the identifier recorded for the activity, and a regular expression for the result code recorded for the activity.

Note The sort of identifier that is recorded for a history record depends completely on the kind of activity the record represents. For a job activity, such as starting a job or ending a job, the identifier will be the job’s identifier. For a document indexing activity, the url of the document will be recorded. For most connector-related activities, the document identifier as defined by the connector will be recorded.

2.5.1 Simple history report

The simple history report is a direct view of the history records you have selected. These records are presented in reverse chronological order initially, so that the most recent activity is first. For our example web job, the history might look something like figure 2.41.

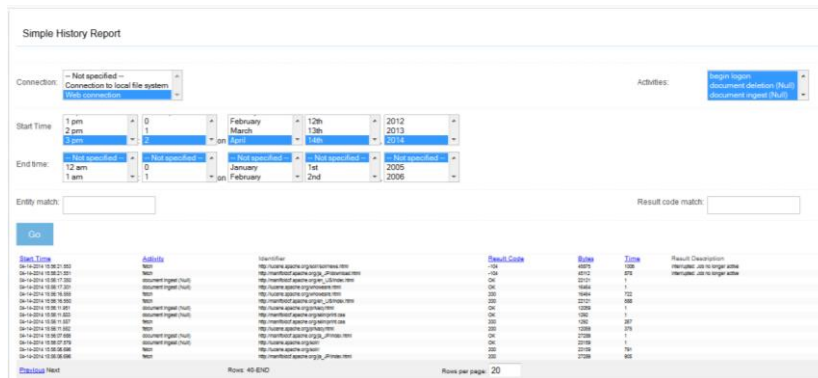


Figure 2.41 A simple history report of a paused web job. Some fetches were interrupted when the job was paused, as you can see by the result code of -104 and the result description that shows that the fetches were interrupted.

The output columns for this report are described in table 2.16.

Table 2.16 Simple history report columns and their meanings.

Column	Meaning
Start time	This is the time the activity started, to the nearest millisecond.
Activity	This is the name of the activity performed.
Identifier	This is a field that describes what entity was involved in the activity. This field's interpretation is dependent on the activity type and on the underlying connector (if any).
Result Code	This is a short code, either numeric or alphabetic, which describes the result of the activity. Once again, this field's interpretation is dependent on the activity type and on the underlying connector.
Bytes	This is the number of bytes involved in the activity.
Time	This is the total elapsed time, in milliseconds, that the activity took.
Result	If the result code is not a successful result code, this field may (or may not)

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

Description	contain an extended description of the problem.
-------------	---

2.5.2 Maximum activity history report

The function of the maximum activity status report is to tally the number of records by class over all time intervals, and calculate the average level of activity per class for that interval. The report is initially sorted so that the class with the interval with the highest average activity is sorted first.

This report has a similarity with the queue status report, in that both reports require you to provide an identifier class description. This field should include a properly parenthesized regular expression, which will extract a class name from every document identifier in the included set. You may also adjust the time interval over which the average is taken from its default of five minutes to something larger, if you are interested in longer term behavior.

The reason this report was originally added was to permit a user to see whether the crawler was operating within the desired throttling restrictions. It is also useful as a means of assessing performance. Typically, for a web job, you would select only one activity (e.g. fetch), and enter a value for the Identifier class description field that extracts the server name from the web URL. In an earlier chapter section, we decided that the regular expression `^https?:\/\/([^\?$]*)` would do this.

If you try such a report on the example web job, you will see something similar to figure 2.42.

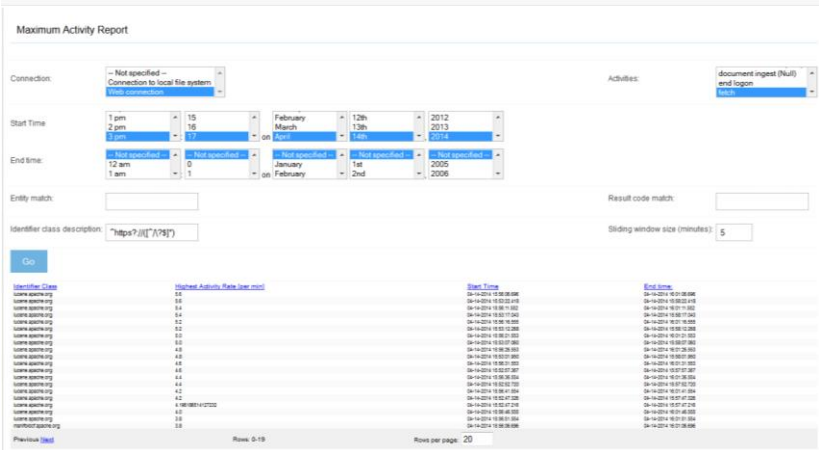


Figure 2.42 A maximum activity report for a web job, run under Apache Derby. The identifier class is repeated, and all time intervals are explicitly listed with their average activity rate. Under PostgreSQL, only the maximum activity rate and interval is listed for each group.

Table 2.17 describes the columns in this report.

Table 2.17 Maximum activity report columns and their meanings.

Column	Meaning
Identifier Class	This is the group name, which corresponds to the parenthesized part of the identifier class description field.
Highest Activity Rate [per min]	This is the maximum number of activity events, per minute, for the group described by the identifier class.
Start Time	This is the start of the time window for the interval that had the maximum activity rate for the group.
End Time	This is the end of the time window for the interval that had the maximum activity rate for the group.

The screen shot clearly shows that the highest activity rate, for a five minute interval, is about 4.6 fetches per minute per server. This is about half the rate we originally set on the `Throttling` tab. So this report actually seems to be useful! But why is there a discrepancy?

The simple answer is “lots of possible reasons”. First, I may not have allowed the job to run even five minutes before I paused it, so it is possible that there’s just not enough of a run to have everything working at top speed. Second, some of the files we saw in the simple history report were quite large, and there is also a bandwidth throttle in place. Perhaps that throttle is providing the critical limit. We can explore that possibility more thoroughly using the maximum bandwidth report, covered in the next section.

Warning Derby’s ability to properly support either the maximum activity or maximum bandwidth report is limited. As of this writing, Derby is unable to perform the step that combines individual records together from a given group into one record that presents the maximum. Therefore, ManifoldCF under Derby presents multiple result rows in situations where the same report under PostgreSQL would present only one.

2.5.3 Maximum bandwidth history report

The maximum bandwidth report is similar in many ways to the maximum activity report, except that rather than counting activities, it counts the bytes involved in those activities instead, to arrive at an average bandwidth, in bytes per second. If you run a maximum bandwidth report on your example web job, making sure to supply an identifier class

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

description regular expression value that will group based on the server name in the URL, you should see a report that looks something like figure 2.43.

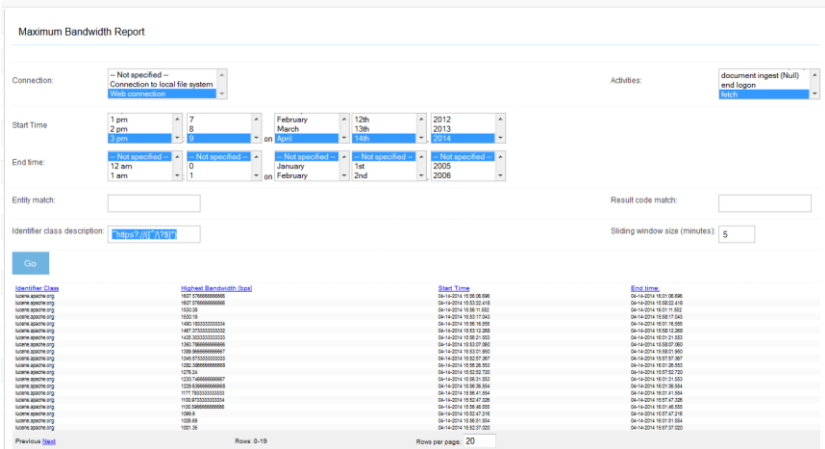


Figure 2.43 A maximum bandwidth report for a web job, run with Derby as the underlying database. The maximum bandwidth for the incubator.apache.org server over a five-minute window is about 10,000 bytes per second.

The columns returned by this report are described in table 2.18.

Table 2.18 Maximum bandwidth report columns and their meanings.

Column	Meaning
Identifier Class	This is the group name, which corresponds to the parenthesized part of the identifier class description field.
Highest Bandwidth [bps]	This is the maximum number of bytes involved in the selected activities, per second, for the group described by the identifier class.
Start Time	This is the start of the time window for the interval that had the maximum bandwidth rate for the group.
End Time	This is the end of the time window for the interval that had the maximum bandwidth rate for the group.

The report above shows that the maximum bandwidth used against a single server in this example is only 10,000 bytes per second, which is one-sixth of the maximum value we set

on the connection definition's `Bandwidth` tab. It appears that bandwidth throttling is also not the limiting factor for the crawl. So, what **is** the limiting factor?

The most likely answer harks back to Chapter 1. Recall that in that chapter we characterized the performance of Derby as too poor for real crawling, and mentioned that it might be a good idea to use PostgreSQL instead. One major reason is how Derby handles deadlocks, and how often it gets into deadlock situations. It turns out that you can get Derby to deadlock against itself with very little effort – all it requires is one thread performing an insert into a table, and another doing a query against the same table at the same time. This would not be a big problem if Derby detected the deadlock situation instantly, but unfortunately the way Derby detects deadlocks is by a timeout mechanism. That means that both of the deadlocking threads must wait 60 seconds before the deadlock is detected. The unfortunate result of these deadlock train-wrecks is a `ManifoldCF` that performs quite poorly.

To confirm this hypothesis, we could actually go back into the simple history and look for time stretches of a minute or so when there is very little activity. I will leave that as an exercise for the student. But the take-away lesson from all this is to use PostgreSQL, if you can.

2.5.4 Result history report

The last history-based report we will cover is the result history report. This report allows you to define groups based on **both** the record identifier **and** the record result code. So, for example, you can obtain a count of fetches per server, and per result code from that server. See figure 2.44.

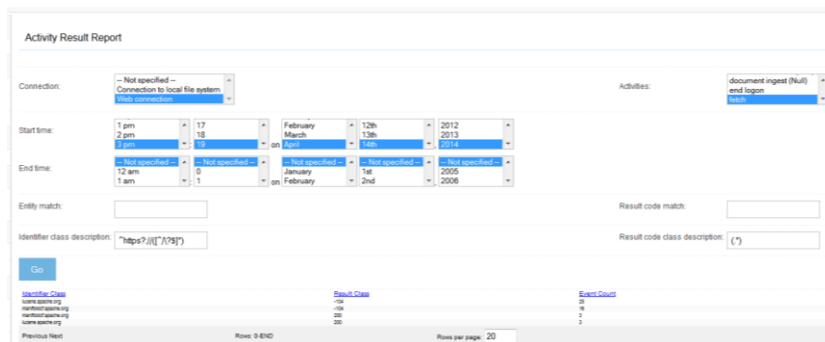


Figure 2.44 A result code report for a web job. The identifier class description groups records by server name, while the result code class description enumerates each result code that is present.

The columns in this report are listed in table 2.19.

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>

Table 2.19 Columns for the result code report, and their meanings.

Column	Meaning
Identifier class	The identifier group, as described by the parentheses in the “Identifier class description” field.
Result class	The result code group, as described by the parentheses in the “Result code class description” field.
Event count	The total number of activity events that took place for all records matching the criteria and matching the identifier class and result class.

This report is particularly useful for figuring out at a glance which sites might be blocking your crawler’s access. In my experience, no matter how careful you are, sooner or later some web master somewhere will decide that they really don’t like you crawling their content. Should that occur, you surely would want to know about it, presumably so that you can beg and plead to get your access to that content restored. The result report can help you keep abreast of any such problems.

Now you have visited all the reports that ManifoldCF provides. I think you will agree that they are likely to be extremely useful in helping to get things set up properly, and in diagnosing problems.

As far as our simple web example is concerned, you may click the job status `Abort` link and shut it down at any time. Or, you may decide to extend our example, and perhaps try to crawl a site that is protected by session authentication. That is actually a very typical kind of setup for a corporate intranet, so many readers will find that such a task resonates well with them. But let me warn you – setting up a crawl to deal with session authentication often requires a great deal of careful analysis and reverse engineering, so this can be a very challenging task. You will find that the on-line documentation for the web connector will help enormously in going about setting up such a crawl, and what you’ve learned in this chapter about the ManifoldCF reports will help you greatly in debugging your session authentication setup.

2.6 Summary

In this chapter we have examined every aspect of the crawler UI, and used it to set up and perform a simple web crawl. We’ve learned how to use the reports, and how to interpret the results. And, we’ve learned quite a bit about web crawls in general while we were at it.

In the next chapter, we will look at a very different way to interact with ManifoldCF. Instead of using the UI, we’ll control ManifoldCF programmatically. While we are at it, we’ll learn quite a bit about a very different way of crawling the web, by using the RSS connector.

Chapter author name: Wright

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=727>