InQuira 6 Application Guide

Developing, Configuring, and Maintaining InQuira 6 Applications

Document Number IQ62-ADG-07

December, 2003

InQuira
851 Traeger Ave
Suite 125
San Bruno, CA 94066

Copyright

Restricted Rights

Trademarks and Service Marks

Contents

Preface

About This Guide

This guide provides information about InQuira 6 application configuration, development and administration. It provides information about configuring the application components, using the configuration and administrative tools, and configuring content processing, the Dictionary, run-time components, and the user interface.

This guide is intended for application developers who want to configure, deploy and maintain a InQuira 6 application. You may also need to consult the InQuira 6 Language Guide in order to complete some Dictionary-related tasks. See ***InQuira 6 Documentation*** on page ix for a complete description of the InQuira 6 product documentation.

This preface includes information on:

- the general organization of this guide
- the support services available from InQuira Customer Support
- the available product documentation and its conventions

What This Guide Contains

The *InQuira 6 Application Guide* is divided into the following sections:

Configuring Content Acquisition **on page 11**

Describes the components that access your organization's content, and how to configure them.

Translating Content to the InQuira Format **on page 55**

Describes the components that translate application content to a standard format, some of which are configurable.

Configuring Access to Business Data **on page 89**

Describes the components that access and store business data obtained from configured data sources, and how to configure them.

Configuring Document Security **on page 107**

Describes how to configure InQuira 6 to permit only authorized users to access answers from secured content.

Scheduling InQuira Processes **see "Scheduling InQuira 6 Processes" on page 115**

Describes how to schedule various InQuira 6 operations by defining and scheduling jobs, using the Scheduler functions of the Configuration and Administration Interface.

Contacting InQuira

You can contact InQuira by mail, telephone, fax, and email.

Mail:	851 Traeger Ave Suite 125 San Bruno, CA 94066
Telephone:	(650) 246-5000
	InQuira Customer Support Hotline: (888) 947-8324
Fax:	(650) 246-5036
Email:	For sales information, send email to **sales@inquira.com**. For product support, send email to **support@inquira.com**.

You can find out more about InQuira on the web at: **www.inquira.com**.

Note: See ***InQuira Customer Support*** on page ix for more information on reporting incidents to InQuira Customer Support.

InQuira Customer Support

InQuira Customer Support is available from 6:30 am to 4:30 pm PST, excluding InQuira holidays.

You can contact InQuira Customer Support by email at: **support@inquira.com**. We recommend that you use email to report all Priority 2, 3, and 4 incidents.

For Priority 1 incidents, please use the support hotline: (888) 947-8324.

Important: We accept Priority 1 requests only by telephone. We recommend that you send a follow-up email for Priority 1 requests after contacting InQuira Customer Support using the support hotline.

Call response times are determined by the following priority definitions:

Priority Level	*Response Time*	*Definition*
1	1 business hour	A production system hangs or crashes, or continued use of the program is impossible.
2	8 business hours	The product is usable with major restrictions on functionality.
3	16 business hours	The product is usable with minor restrictions on functionality.
4	3 business days	You have a question or an enhancement request pertaining to the software or the documentation.

InQuira 6 Documentation

InQuira 6 is shipped with the following documentation set. Each document in the set contains a specific type of information to help you use the product.

Document	*Number*	*Description*
InQuira 6 Installation Guide	IQ62-IG-07	This guide is intended for technical staff who are responsible for installing InQuira 6. It provides detailed information on installing and configuring InQuira 6 products and components.

InQuira 6 Application Guide	IQ62-ADG-07	This guide is intended for application developers who need to develop and deploy an InQuira 6 application. It describes InQuira 6 integration, development, configuration, and maintenance.
InQuira 6 Language Guide	IQ62-LDG-07	This guide is intended for application developers and subject matter experts who need to create, implement, and maintain aspects of the InQuira 6 Dictionary. It provides an overview of the Dictionary components, and describes customization, maintenance, and administration tasks and the tools, processes, and procedures required to perform them.
InQuira 6 User Interface Guide	IQ62-UIG-07	This guide is intended for application developers who want to integrate and customize the InQuira 6 Dynamic Portal User Interface. It contains information about the elements and features of the User Interface, and provides guidelines for integrating it into your web architecture, customizing its appearance and functionality, and implementing various special features.
InQuira 6 Analytics Guide	IQ62-AG-07	This guide is intended for application developers who want to configure, deploy, and maintain InQuira 6 Analytics, and for business analysts who want to use InQuira 6 Analytics to report on InQuira 6 performance.

Documentation Delivery

InQuira 6 documentation is distributed with the software release as a set of Portable Document Format (PDF) files. InQuira documentation is available only to licensed users of our software products and may not be redistributed in any form without express permission from InQuira, Inc.

Note: You need the Adobe Acrobat reader to view PDF documents. The Adobe Acrobat reader is available from Adobe Systems at: **http://www.adobe.com**.

If you encounter a problem, need help using the documentation, or want to report an error in the content, please contact InQuira Customer Support, as described in ***InQuira Customer Support*** on page ix.

If you need help obtaining InQuira product documentation, or want to obtain permission to redistribute a portion of the contents, contact your InQuira account representative.

Documentation Conventions

We use the following typographical conventions in our documentation:

Convention	Definition
monospace	We use monospace font to denote code examples, path and file names, parameter values, and system messages and prompts.
italics	Italics indicate terms contained in the glossary and citations of other published sources. Strings of italicized text within file names or parameters indicate variable text that you must replace with an appropriate string, or that the system will replace with product- or installation-specific values. Product directories may use italicized *n* characters as variables to denote current product version numbers.
<value>	Indicates a symbolic for which you must specify an appropriate value. For example, `<section>` indicates a specified section of the answer display page.
%	Indicates the Unix command-shell prompt without root privilege.
C:	Indicates the Microsoft Windows command prompt or file system.
[item]	Indicates an optional item within a syntax description.
{item}	Indicates a required item within a syntax description, only when necessary for clarity. Otherwise, required items appear without braces.
|	Separates choices in lists of optional and required items within syntax descriptions. For example, `[item1|item2]` or `{item1|item2}`.
$TERM or **${TERM}**	Indicates a variable (within Unix-type environments) for which you specify an appropriate value. For example, $ROOT indicates the root directory of the installed product. Variables are sometimes enclosed in curved braces when necessary for clarity. In some instances, variables are set by and/or resolved by the system with no user intervention required.
%TERM%	Indicates a variable (within Microsoft Windows environments) for which you specify an appropriate value.

Screen and Text Representations

The depictions of product screens, screen text, and file contents in our documentation are representations. We attempt to accurately convey the product's appearance and functionality; however, actual product contents and displays may differ from the published examples.

Chapter 1

Introduction to Application Development

This section describes the tasks involved in the InQuira 6 application development process and provides an overview of the major components that developers will work with, and the environments in which these components will be configured and deployed.

InQuira 6 Components

InQuira 6 is installed with the following components that you configure for use in your environment:

The Natural Interaction Engine

The Natural Interaction Engine provides the run-time request-response processing functions and interacts with the other major components at run-time.

The Content Processor

The Content Processor makes your site content available to the Natural Interaction Engine in the form of multiple indices, which encode semantic and location information for each processed document.

The Content Processor includes the content acquisition crawlers, the preprocessor, which translates your site's content into a standard XML format, and the indexer.

The Dictionary

The Dictionary contains the Ontology, which consist of the hierarchical representations of semantic relationships that define words and phrases in your application. It also contains the Rules, which is a set of expressions that encode the desired interpretation and actions associated with end-user requests and application content.

The Rules Engine

The Rules Engine operates at run-time to compare user requests with the set of Rules in the Dictionary. Each Rule that is true for a request generates an action to perform, ulitimately resulting in the best possible response for the request being processed.

Workbench and Configuration and Administration Interface

InQuira 6 provides a complete set of application configuration, management, and analysis tools, including:

- the Workbench, which contains the Dictionary Manager, user management, application testing, and Analytics administration interfaces
- the Configuration and Administration Interface, which provides access to the application configuration and administration functions, including content processing and user interface configuration.

Chapter 2

InQuira 6 Architecture

InQuira 6 architecture is based on components that run as services. The various InQuira 6 services respond to requests to send and recieve data from other services.

A InQuira 6 service is a part of a InQuira 6 application and/or a InQuira 6 instance. An application is a collection of configured services that are required to perform the fundamental content and request/response processes. An instance is a collection of configured services, including a configured application, that runs within a single Java Virtual Machine (JVM).

The InQuira 6 services are described in ***InQuira 6 Services*** on page 3.

The InQuira 6 installation process results in a configured default application and configured default instance within the installation location. You can use the default application and the default instance to begin working with InQuira 6.

InQuira 6 Services

InQuira 6 architecture is based on components that run as services. Services are long-running processes. The services respond to requests to send and recieve data from other services.

A service is a part of an application and/or an instance.

A service's behavior is determined by its configuration.

Chapter 3

Configuring the Content Store

The InQuira 6 Content Store is a database-backed repository of application content. You configure the content store database using the Configuration and Administration Interface.

Configuring the Content Store Data Source

To configure the data source for the content store,

- select General Settings from the Configuration and Administration Interface

The General Settings page displays.

- select Edit List in the Content Store Data Source field

The Editing:Content > Data Sources page displays

- select an existing item, or Add New Item
- specify the data source connection parameters, as described in ***Specifying Data Source Connection Parameters*** on page 5

Specifying Data Source Connection Parameters

Specify the following parameters to connect the InQuira 6 application to a data source.

URL

Enter the connection URL for the JDBC connection. The value of the connection URL depends on your RDBMS and JDBC driver.

Driver Class Name

Enter the fully qualified JDBC driver name.

Exception Handler

The exception handler checks for known error conditions, such as unique key violations. You can configure one or more of the appropriate pre-defined exception handlers for re-use in multiple application configurations. InQuira 6 supplies the following pre-defined exception handlers:

- Oracle 9i
- Microsoft SQLServer 2000
- MySQL 3

Connection Properties

Specify any connection properties, such as user name and password, as required.

Chapter 4

Configuring the Dictionary

You configure the Dictionary and the Dictionary Manager for use with your application through the Configuration and Administration Interface. See *InQuira 6 Language Guide* for information on the structure and contents of the Dictionary and information on using the Dictionary Manager.

You configure the application Dictionary by specifying:

- the local directory that stores the Dictionary
- the connection to the remote Dictionary repository
-

You can also configure:

- the domains that the Dictionary will use
- settings for default answer priorties and answer purposes in the Dictionary Manager
- the initial node of the concept hierarchy to display in the Dictionary Manager

Specifying the Local Directory for the Dictionary

The local Dictionary directory stores the Dictionary objects for use by the application and the Dictionary Manager. To configure the local directory for the Dictionary:

- select Dictionary Manager from the Configuration and Administration Interface

The Dictionary page displays.

- specify the fully qualified directory name. For example, the local directory:

```
C:/inquira/repostitory_name
```

Configuring the Ontology

The CVS root string specifies the connection to the remote repository that contains the Dictionary objects. To specify the CVS root:

- select Dictionary Manager from the Configuration and Administration Interface

The Dictionary page displays.

- specify the CVS root as in the example format:

```
:pserver:user_name@machine_alias:/path_to_cvs_root_dir
```

Consult the CVS documentation, available at **http://www.cvshome.org** for more information.

Specifying the Root Concepts for the Dictionary Manager

The Dictionary Manager Concept Tree window displays hierarchies of Concept *type of* or *part of* relationships defined in the Dictionary. The hierarchies originate at Concepts that you specifiy as *Root Concepts* in the application configuration.

The Dictionary Manager requires at least one Root Concept to be defined in the application configuration in order to display the Concept tree. You can configure one or more Root Concepts using the Workbench page of the Configuration and Administration Interface.

To add or change a Root Concept in the Configuration and Administration Interface:

- select Workbench from the Application Instance area of the Configuration and Administration Interface main menu

The Workbench page lists any currently configured Root Concepts. The order of the list does not affect the Concept Tree display.

- select Edit

The Editing: Workbench page displays the current Root Concepts as selectable items, and an Add New Item option.

- select Add New Item, or select an existing Root Concept to modify, if necessary

The Editing: Workbench > Root Concepts page displays an editable field to contain concept information.

- specify the desired Root Concept, using the Concept name format:

```
part_of_speech.domain_name:concept_headword
```

Note: See the *InQuira 6 Language Guide* for more information on Concept naming conventions.

- select OK to save your changes

Chapter 5

Configuring Content Acquisition

InQuira 6 responds to end-user requests by providing answers derived from your organization's content. InQuira 6 acquires content from a variety of sources, including unstructured documents and structured data.

InQuira 6 acquires unstructured content using its content processing system to collect, translate, and index unstructured text for use by the request and response processing components, as described in ***Acquiring Unstructured Content*** on page 13.

InQuira 6 acquires structured data by connecting directly to the source database, issuing user requests as SQL queries, and presenting the results, as described in ***Accessing Structured Content*** on page 44.

You can also configure the application to use both the structured data and unstructured data retrieval modules to return answers from text fields within database content.

The Content Acquisition Process

The content acquisition process is the first step in content processing. Content acquisition makes your organization's content available to the application for preprocessing and indexing. The content acquisition system comprises components that:

- copy content from configured servers, file systems, databases, and other repositories
- store the content in a compressed form within the Content Store
- provide content to the Preprocessor, which converts the content to a standardized format for use by other content processing components, as described in ***Translating Content to the InQuira Format*** on page 55.

The components that make up the content acquisition system are described in ***Content Acquisition Components*** on page 12.

You can configure content acquisition to access content from multiple servers, file systems, databases, and other repositories, as described in ***Acquiring Unstructured Content*** on page 13 and ***Accessing Structured Content*** on page 44.

You use the content acquisition system by configuring various content crawler functions to access your organization's content repositories, as described in ***Configuring Content Crawlers*** on page 13.

Note: You must have a configured Content Store available to use the content acquisition process. See ***Configuring the Content Store*** on page 5.

Content Acquisition Components

The content acquisition system contains the following components:

Content crawlers

Content crawlers access content stored on various types of servers, file systems, and in other repositories. Content crawlers traverse specified directories or repositories and copy eligible file contents to the Content Store according to scheduling and collection criteria that you specify. Each scheduled crawler task launches an instance of a specific crawler configuration. Each unique crawler configuration defines a document collection.

There are several types of content crawlers, each designed to access a particular type of file system, server, or repository:

- Web
- News
- File
- Database

Document collections

Document collections are logical entities defined by unique crawler configurations. Each unique crawler configuration defines a document collection. See ***Specifying Collections*** on page 14.

Document filters and file filters

Document filters and file filters define inclusion criteria for directories, documents, or files. See ***Configuring Content Acquisition Filters*** on page 14.

Document attributes

Document attributes assign metadata that can be used to restrict information retrieval during request processing. See ***Configuring Document Attributes*** on page 17.

Document supertitles

Document supertitles assign metadata to create logical groups or categories of documents. See ***Configuring Document Supertitles*** on page 18.

Acquiring Unstructured Content

You can access unstructured content by configuring and scheduling one or more unstructured content crawlers, as appropriate. You can specify multiple configurations of a single crawler type to define various collections of unstructured content.

Each unique crawler configuration defines a collection, and each document can belong to only one collection. See ***Specifying Collections*** on page 14 for more information.

InQuira 6 provides the following content crawler types to access unstructured content:

Web crawler

Accesses content stored on HTTP (web) servers. See ***Configuring Web Servers*** on page 22.

News crawler

Accesses content stored on NNTP (newsgroup) servers. See ***Configuring Newsgroup Servers*** on page 31.

File crawler

Accesses content stored on file systems. See ***Configuring File Systems*** on page 33.

Database crawler

Accesses unstructured content stored within databases. See ***Configuring Content Acquisition from Databases*** on page 34

See ***Accessing Structured Content*** on page 44 for information on accessing structured content from databases.

Configuring Content Crawlers

You configure content crawlers using the Content configuration pages in the Configuration and Administration Interface. Each unique crawler configuration defines a document collection. You specify crawler configuration values to:

- define the file types and locations of the content to access
- assign attributes and supertitles to documents within the collection
- define the full document retrieval URLs that the User Interface will display with the answers derived from the collection

You define as many configurations of each type of crawler (document collections) as you need to account for the different types of documents and data that you want to include in content processing.

Specifying Collections

A document collecton designates a logical set of documents. You define a document collection by specifying a set of crawler configuration parameters and assigning a collection name to the set.

You specify document collections for configurations of all crawler types. Each crawler type is designed to access one type of server, file system or repository; however, each source of content may contain multiple document types, each having different acquisition requirements. You can define multiple configurations of a single crawler type to tailor content acquisition for each logical set of documents that you identify.

You specify document collections using the Collection Name field on the crawler configuration page. You can specify any unique alphnumeric string. Spaces and punctuation are not valid.

Specifying Variance Thresholds for Collections

You can specify thresholds to limit the amount of variance that the application will allow in the number of documents to process. You can specify both upper and lower thresholds for the acquisition (crawl) process, preprocessing, and the indexing process.

You specify the variance thresholds on the Content page of the Configuration and Administration Interface. You specify the threshold as a floating point value between 0 and 1. The application converts this value to a percentage, and compares the specified threshold values to a baseline that is established for each collection.

The baseline is set by the first accepted crawl. You can reset the baseline using a predefined task in the administration scheduler, as described in ***Content Acquisition Tasks*** on page 120.

For example, consider a collection containing 700 documents at its first crawl. The application sets the baseline for the collection at 700.

If the collection has acquisition upper and lower limits set to 0.1 (10%) and 0.2 (20%) respectively, then the application will terminate acquisition processing if a crawl yields more than 770 or less than 560 documents (700 plus 10% = 770 ; 700 - 20% = 560).

Configuring Content Acquisition Filters

Content acquisition filters determine the directories, files, and documents that a crawler will access during content acquisition. Content acquisition filters specify criteria for inclusion based on filename and document metadata. You can specify the following types of content acquisition filters within crawler configurations:

- ***Configuring File Filters*** on page 15
- ***Configuring Document Filters*** on page 17

Configuring File Filters

File filters are logical statements that determine which directories and files within a file system will be included in content acquisition. You specify file filters only for File crawlers.

File filters specify criteria for processing file objects (directories and files within directories) based on specified filter patterns that the crawler compares to the file object name. You can configure file filters to perform the following content processing operations:

- include the file object in content processing
- exclude the file object from content processing
- include the file object in the Content Store, but exclude it from content processing

File filters have both a purpose, as described in ***File Filter Purposes*** on page 15, and a type, as described in ***File Filter Types*** on page 15.

File Filter Purposes

File filter purposes determine which file objects the filter will evaluate:

File Patterns

File patterns evaluate file names. The crawler will include the content of accepted files.

Directory Patterns

Directory patterns evaluate directory names. The crawler will evaluate the files and sub-directories within accepted directories. The crawler will not evaluate the files or sub-directories within rejected directories.

Recurse Patterns

Recurse patterns evaluate directory names. The crawler will evaluate the files within sub-directories of a rejected directory. The crawler will not evaluate the files within rejected directories.

Process File Patterns

Process file patterns evaluate file names. The crawler will include the content of the files in the Content Store, but will exclude them from preprocessing and indexing. This content will not be available for request processing.

File Filter Types

You can specify the following types of patterns for each filter purpose:

- accept patterns, which must be true for a file to be eligible

- reject patterns, which must be false for a file object to be eligible

You can specify multiple patterns within a single file filter.

As the crawler traverses a file system, it tests each file object's name against the accept and reject patterns for all of the specified filter purposes. The crawler eliminates file objects for which an accept pattern is false, then additionally eliminates file objects for which a reject pattern is true.

Specifying File Filters

You configure a File filter by specifying the Filter name in the crawler configuration. You can specify multiple File filters within a crawler configurattion.

You can specify pre-defined filters included in InQuira 6, and custom filters that you define. To define a custom File filter:

- specify a filter name. The filter name can be any alphanumeric string. Punctuation and spaces are not allowed.
- specify one or more filter patterns, as described in ***Defining Filter Patterns*** on page 16

Defining Filter Patterns

Filter patterns are logical statements within file filters. Filter patterns specify criteria that the crawler compares to the file object name to determine processing eligibility. You can specify multiple filter patterns within a file filter.

You define a filter pattern by specifying a regular expression that resolves to a state of true or false when the crawler compares it to a file object name. Valid filter pattern expressions must conform to the Java regular expression standard, `java.util.regexp`. See `http://java.sun.com` for more information.

You define filter patterns on the Editing: Content > File Crawlers > File Filter > Simple File Filter > Simple File Filter page of the Configuration and Administration Interface.

To define a filter pattern:

- enter a regular expression to define the accept or reject criteria for the filter purpose

Note: You can view examples of valid pattern expressions by selecting pre-defined filter patterns from the Configuration and Administration Interface pages.

Configuring Document Filters

Document filters are logical statements that determine which documents will be included in content acquisition. You can specify document filters for News, File, Database, and Custom crawlers.

Document filters specify one or more conditions for inclusion based on document metadata. The crawler tests the conditions against each document's metadata, and accepts only those documents for which all conditions are true.

Specifying Document Filters

You specify Document filters on the Editing: Content > File Crawlers > Document Filter > Simple Document Filter > Simple Document Filter page of the Configuration and Administration Interface. You can specify any number of Document filters.

Each document filter consists of a name and one or more document conditions.

To specify a Document filter:

- specify a Document filter name. The name can be any alphanumeric string. Punctuation and spaces are not allowed.
- specify a Document condition, as described in ***Specifying Document Conditions*** on page 20

Configuring Document Attributes

Document attributes are metadata that you assign to restrict information retrieval during request processing. You can define document attributes that correspond to content purpose, such as marketing collateral. You specify document attributes for Web, News, File, and Database crawlers.

You assign document attributes by specifying one or more document attribute selectors in a crawler configuration. Each document attribute selector specifies one or more document attributes and one or more document conditions.

The crawler assigns the specified attributes only to documents for which the specified conditions are true. All specified conditions must be evaluated as true, and all specified attributes are assigned to the document. The assigned attributes are then part of the metadata associated with documents within the Content Store.

Note: You can modify attributes for documents within an existing Content Store by updating attributes in the application configuration and scheduling an attribute update task.

Document Attribute Selectors

Document Attribute Selectors are the mechanism for specifying document attributes. Each document attribute selector specifies:

- one or more document attributes
- one or more document conditions

You can define document attributes that reflect application-specific information retrieval preferences, or configure relevant document attributes defined in an external application, such as a CRM package.

Document conditions are logical statements made up of a filter item, and comparator, and a comparison value, as described in ***Specifying Document Conditions*** on page 20.

Specifying Document Attributes

You can specify one or more document attributes within a document attribute selector. You specify document attributes using the Document Attributes page of the Configuration and Administration Interface. You specify an attribute name and whether the attribute is permanent. Permanent attributes reference document metadata from external sources, for example, an external content management system. Permanent attributes are retained when you use the application scheduler function to update document attributes.

To specify a document attribute:

- select Document Attribute from the Document Attribute Selector page
- enter the attribute value. You can specify only one attribute per entry. Attributes can be any alphanumeric string. Spaces and punctuation are not allowed.
- specify permanent status, if desired, by selecting the Permanent radio button

Configuring Document Supertitles

Document supertitles are metadata that you assign to create logical semantic groupings of documents. You can specify document supertitles for News, File, Web, and Database crawlers.

You assign document supertitles by specifying one or more document supertitle selectors in a crawler configuration. Each document supertitle selector specifies one or more document supertitles and one or more document conditions.

The crawler assigns the specified supertitles only to documents for which the specified conditions are true. All specified conditions must be evaluated as true, and all specified supertitles are assigned to the document. The assigned supertitles are then part of the metadata associated with documents within the Content Store.

Note: You can modify supertitles for documents within an existing Content Store by updating supertitle information in the application configuration and scheduling a supertitle update task.

Document Supertitle Selectors

Document Supertitle Selectors are the mechanism for specifying document supertitles. Each document supertitle selector specifies:

- one or more document supertitles
- one or more document conditions

Document supertitles are user-defined values. You can define document supertitles that correspond to semantic categories, for example, product lines or release levels that may not be explicit in the document contents.

Document conditions are logical statements made up of a filter item, and comparator, and a user-defined comparison value. See ***Specifying Document Conditions*** on page 20 for a complete description of document conditions.

Specifying Document Supertitles

You can specify one or more document supertitles within a document supertitle selector. You specify document supertitles using the Document Supertitles page of the Configuration and Administration Interface.

To specify a Document Supertitle:

- select Document Supertitle from the Document Supertitle Selector page
- enter the supertitle value. You can specify only one supertitle per entry. Supertitles can be any alphanumeric string.

Specifying Document Retrieval URLs

The User Interface presents formatted responses that include links to the full documents from which the responses were derived. For file systems, the web server URL is typically different from the location that the application uses to access the content for processing.

You specify document retrieval URLs to ensure that the application can create correct URLs for display within the User Interface.

You specify document retrieval URLs for File crawlers.

You configure the retrieval URL by specifying values for the segments of the URL that correspond to:

- host
- port
- prefix

The application then creates the full display URL using additional document-specific information.

To specify a full document retrieval URL for a collection:

- select File Crawler from the Editing:Content page

The Editing:Content > File Crawler page displays.

- specify values for the URL segments in the form:

```
host.port.prefix
```

Specifying Document Type Detectors

The Document Type detector specifies the type or types of documents that a crawler is intended to access. The document type detector operates by comparing the file extensions of candidate files to the values specified in its configuration.

You specify document type detectors for File and Web crawlers. You specify a single Document Type Detector for a crawler configuration.

Each Document Type Detector specifies one or more mappings of a document type name to one or more file extensions. For example, a single document type detector called `Web` might specify two types of web documents, `HTML` and `PDF`. The mappings for this detector might be:

Document Type	*Mapped Extensions*

HTML	`html`
	`htm`
	`HTML`
	`HTM`

PDF	`pdf` `PDF`

Specifying Document Conditions

Document conditions are logical statements that you specify within document attribute selectors, as described in ***Configuring Document Attributes*** on page 17, and document supertitle

selectors, as described in ***Configuring Document Supertitles*** on page 18. Crawlers use conditions to determine whether an attribute or supertitle will be assigned to a document.

Crawlers determine attribute and supertitle assignments by comparing existing document metadata to all of the conditions specified in the configured document attribute selector or document supertitle selector. Crawlers assign attributes and supertitles only if all specified conditions are true.

Document conditions contain:

- a filter item
- a comparator
- a comparison value

You specify these elements to define a document condition in the form:

```
filter_item, comparator, comparison_value
```

where:

filter_item specifies a defined filter item that corresponds to available document metadata. See ***Specifying Filter Items*** on page 21.

comparator specifies a boolean comparator. See ***Specifying Comparators*** on page 22.

comparison_value specifies a user-defined comparison value. See ***Specifying Comparison Values*** on page 22.

For example, you might set a condition to exclude empty documents by specifing a `document size` filter item, a comparator of `> (greater than)`, and a comparison value of `0`.

You define document conditions using the Editing: Content > File Crawlers > Document Attribute Selector > Conditions page of the Configuration and Administration Interface.

Specifying Filter Items

Filter items are the basis of comparison for document conditions, document filters, and file filters within crawler configurations. In document filters and file filters, filter items provide direct matching based on document metadata. In document conditions, filter items are parts of logical statements that also include boolean comparators and user-defined comparison values.

Filter items consist of:

- a name, which is a simple descriptive label
- a method, which provides the means of accessing and matching the required metadata

The following filter items are available in InQuira 6:

- getDisplayURL
- getDocumentSize
- getLastModificationTime

Specifying Comparators

Comparators are boolean operators within document conditions. Comparators specify the basis of comparison between the specified filter item and the user-defined comparison value.

The valid comparators that you can specify within document conditions are:

>	greater than
<	less than
=	equals
!	not
~	matches a regular expression

You can combine these basic elements to define more complex comparators. For example, you can specify >= to indicate `greater than or equal to`.

Specifying Comparison Values

Comparison values are user-defined elements within document conditions. Comparison values specify the value that the document metadata will be compared to when the crawler evaluates the document condition. For example, in a condition to exclude empty documents, the comparison value of `0` specifies the value for document size metadata comparison.

Configuring Web Servers

You configure content acquisition from web (HTTP) servers by configuring and scheduling one or more Web crawlers to access the desired directories and copy new or modified content into the application.

You can configure multiple Web crawlers to access:

- documents located on multiple HTTP servers
- documents on a single HTTP server that have different collection requirements

Each unique crawler configuration defines a document collection. Each configured collection specifies parameters and rules that determine:

- general crawler parameters, such as the collection name and document types to process
- the scope of the crawl, including the starting point and the crawl depth
- the documents to download from crawled locations
- the downloded documents to process as content for the application

You specify Web crawler parameters using the Editing: Content > Web Crawlers page of the Configuration and Administration Interface.

The Web Crawl Process

The Web crawler runs as a scheduled job that you can administer using the scheduling function, as described in ***Scheduling InQuira Processes*** see "Scheduling InQuira 6 Processes" on page 115. The Web crawler locates URLs according to the starting points and other crawling scope parameters that you specify as described in ***Specifying the Scope of the Crawl*** on page 25.

The crawler downloads documents according to a set of URL pattern matching rules that you specify as described in ***Specifying Documents for Downloading*** on page 29. It then evaluates the list of downloaded documents against an optional set of processing rules, as described in ***Specifying Documents for Processing*** on page 30, to determine whether to include each document in the application content.

Specifying General Crawler Parameters

You specify the following general crawler collection parameters to:

- define the collection to the application, as described in ***Specifying Crawler Definition Parameters*** on page 23
- define connection values, as described in ***Specifying Crawler Connection Parameters*** on page 24
- control crawler traffic on the web server, as described in ***Specifying Crawler Traffic Parameters*** on page 25

Specifying Crawler Definition Parameters

You can specify the following crawler definition parameters on the Editing: Content > Web Crawlers page of the Configuration and Administration Interface.

To specify crawler definition parameters:

- select Basic Options from the dropdown menu
- specify values for the following parameters:

Available for Unstructured Search	specifies whether the documents in this collection will be available to the unstructured information retrieval module. This parameter is required. Valid values are `On` and `Off`. `On` is the default.
Document Type Detector	specifies a document type detector for the crawler. This parameter is required. Valid values are defined document type detectors, as described in ***Specifying Document Type Detectors*** on page 20
Document Attribute Selector	specifies an optional document attribute selector for the crawler. Valid values are defined document attribute selectors, as described in ***Configuring Document Attributes*** on page 17
Document Supertitle Selector	specifies an optional document supertitle selector for the crawler. Valid values are defined document supertitle selectors, as described in ***Configuring Document Supertitles*** on page 18

- select OK to save the specified values in your configuration.

Specifying Crawler Connection Parameters

You can specify the following crawler connection parameters on the Editing: Content > Web Crawlers page of the Configuration and Administration Interface.

To specify crawler definition parameters:

- select Basic Options from the dropdown menu
- specify values for the following parameters:

Proxy Host	specifies the host name of a proxy server, if required. This is an optional parameter.
Proxy Port	specifies the port to connect to a proxy server, if required. This is an optional parameter.

- select Advanced Options from the dropdown menu
- specify values for the following parameters:

Start Referer	specifies a value to use as the refering page for the crawler starting point. This parameter is required. Valid values are any string. The default is –.
User Agent	specifies a user agent or requestor name to use as input to the crawled site. This field is optional.

- select OK to save the specified values in your configuration

Specifying Crawler Traffic Parameters

You can specify the following crawler traffic parameters on the Editing: Content > Web Crawlers page of the Configuration and Administration Interface.

To specify crawler traffic parameters:

- select Advanced Options from the dropdown menu
- specify values for the following parameters:

Sleep Time	specifies a time interval in seconds that the crawler will wait between requests. This parameter is required. Valid values are any integers. The default is 0.
Bandwidth Throttle	specifies a value to limit the amount of crawler traffic to and from the server. Specify a value in `bytes per second`, for example `1024`. This parameter is optional. There is no default.

- select OK to save the specified values in your configuration.

Specifying the Scope of the Crawl

You specify the scope of the crawl within the structure of the Web server by specifying the following parameters on the Editing: Content > Web Crawlers page of the Configuration and Administration Interface.

To specify crawl scope parameters:

- select Basic Options from the dropdown menu
- specify values for the following parameters:

Ignore Robot Config — specifies whether the crawler will ignore `robots.txt` files and continue crawling as specified by the specified crawler configuration. Valid values are `On` and `Off`. `On` is the default.

Use Cookies — specifies that the crawler will accept cookies from the site. This parameter is required. Valid values are `On` and `Off`. `On` is the default.

Form Fields — specifies optional values to use as input for required form fields within crawled pages, as described in ***Specifying Form Field Values*** on page 27.

Starting Points — specifies the starting points for the crawl process. You can specify mulitple starting points for a single crawler configuration. This parameter is required. Valid values are valid URLs. There is no default value.

- select Advanced Options from the dropdown menu
- specify values for the following parameters:

Max Crawl Depth — specifies the maximum number of link levels to access during the crawl process. This parameter is required. Valid values are any integers. The default is `1000`.

Other Hosts — specifies whether to crawl hosts that do not match the first start point entered. This parameter must be set to `On` to enable a single crawler configuration to access multiple hosts. This parameter is required. Valid values are `On` and `Off`. `Off` is the default.

Entire Host — specifies whether to download everything from the host that is located belowthe specified starting point (`Off`), or to download everything from the specified host, including refered pages that are above the specified starting point in the directory structure.

For example, the starting point `www.inquira.com/customers` may contain a link to `www.inquira.com`. If this parameter is `On`, the crawler will download the page. If `Off`, the crawler will not download the page.

This parameter is required. Valid values are `On` and `Off`. `On` is the default.

Flexible Host Name	specifies whether to download any URL in the domain, or to discriminate between variants, for example, `www.inquira.com` and `ww2.inquira.com`. This parameter is optional.
Visit Many List	specifies an optional list of one or more URLs that the crawler can access multiple times within a single crawl. This feature supports content access for pages that can vary over multiple crawls due to context dependencies, such as having a different cookie set in the crawler. The crawler will consolidate duplicates and only download one copy of pages that are not listed by this parameter.
Additional Allowed URLs	specifies an optional list of additional URLs that the crawler can access from within the specified domain.

- select OK to save the specified values in your configuration

Specifying Form Field Values

You can specify optional values to use as input for required form fields within crawled pages. You specify Form Field values on the Editing: Content > Web Crawlers page of the Configuration and Administration Interface.

To specify form fields:

- select Add New Item under Form Fields

The Editing: Content > Web Crawlers > Form Fields page displays.

- specify the Form Action URL field as follows:

Form Action URL	specifies a list of one or more document URLs to apply form field values to

- select Add New Item under Form Fields

The Editing: Content > Web Crawlers > Form Fields > Form Fields page displays.

- specify the following parameters:

Form Field Name	specifies the HTML field name to apply the Form Field Value to. This is a required parameter for the Form Field feature. Valid values are: • text • password • hidden • textarea • checkbox • radio • select/option • submit For more information on HTML form fields, refer to the World Wide Web Consortium web site: `http://www.w3.org.`
Form Field Value	specifies the value to enter for the specified Form Field Name. This is a required parameter for the Form Field feature.

- select OK to save the specified values in your configuration

Specifying Starting Points

HTTP crawler starting points are URLs that define starting points for crawler configurations. You can configure multiple starting points for a single crawler.

You specify starting points using only the pathname portion of the URL. Do not specify the protocol portion, which is assumed to be `http://`.

Note: You cannot configure the HTTP crawler for use with other URL types, such as FTP. Instead, configure a custom crawler, as described in ***Creating Custom Content Crawlers*** on page 36.

You can add, modify, or delete HTTP crawler starting points.

To add or modify a starting point:

- select Crawlers from the Content Acquisition area of the Configuration and Administration Interface menu

The Content page displays.

- select Edit, then select a new or existing Web Crawler

The Editing: Content > Web Crawlers page displays.

- edit an existing URL as appropriate

or

- select Add New Item in the Starting Points field to add a new starting point

The Editing: Content > Web Crawlers > Starting Points page displays the URL field.

- enter a URL
- select OK to save the specified starting points

You can delete a starting point using the delete [X] icon to the right of the selected item.

Specifying Documents for Downloading

You specify which documents to download from locations within the scope of the crawl by specifying Download Pattern Match rules. For example, you can specify a Download Pattern Match rule to exclude image files from downloading.

Download Pattern Match rules include and exclude documents based on regular expression patterns that you specify, as described in ***Specifying Download Pattern Match Rules*** on page 30. You can specify multiple patterns to define inclusion and exclusion criteria. You specify inclusion or exclusion patterns as regular expressions, using the standard regular expression syntax as supported by the Java regular expression standard, `java.util.regexp`. See the Java Technology Home Page at `http://java.sun.com` for more information.

The crawler processes the specified patterns to determine download status for each document that it locates. The crawler uses a last-match processing rule; the last regular expression rule that applies to a given URL will determine its download status. The crawler will wildcard match around the pattern included in the expression by default. For example, the expression:

```
www\.inquira\.com
```

will match any URL that contains the string `www.inquira.com`, even a query string; however, the expressions

```
http://www\.inquira\.com
https://www\.inquira\.com
```

restrict the match to the specified domains.

Note: The expression `\.` escapes the period (`.`) character, which otherwise matches exactly one character.

Specifying Download Pattern Match Rules

You specify Download Pattern Match rules on the Editing: Content > Web Crawlers page of the Configuration and Administration Interface.

To specify a Download Pattern Match rule:

- select Add New Item under Download Pattern Match

The Editing: Content > Web Crawlers > Download Pattern Match page displays.

- specify a regular expression in the Regex Pattern field
- select the On or Off radio button in the Allow field to specify whether the crawler will include (`On`) or exclude (`Off`) URLs that match this pattern
- select OK to save the specified values to your configuration

Specifying Documents for Processing

Process rules specify whether to process a document that has been downloaded by the crawler. The crawler downloads documents according to Download Pattern Match rules, as described in ***Specifying Documents for Downloading*** on page 29.

The Process rules select downloaded documents for processing based on specified filters, as described in ***Specifying Process Rules*** on page 30.

Specifying Process Rules

You specify Process rules on the Editing: Content > Web Crawlers page of the Configuration and Administration Interface.

To specify a Process rule:

- select Add New Item under Process Rule

The Editing: Content > Web Crawlers > Process Rule page displays.

- select the On or Off radio button in the Allow field to specify whether the crawler will include (`On`) or exclude (`Off`) documents that meet the specified filter criteria
- specify the following parameters to define document filters:

Mime Type	specifies a media type or content type, which denotes the nature of the document. Valid MIME types are specified in RFC 2045 and RFC 2046 of the HTML specification. See the World Wide Web Consortium home page at `http://www.w3c.org` for more information.
Mime Subtype	specifies a media content subtype, which denotes the nature of the document. See the World Wide Web Consortium home page at `http://www.w3c.org` for more information.
Minimum Document Size	specifies a minimum size, in bytes
Maximum Document Size	specifies a maximum size, in bytes

Configuring Newsgroup Servers

You configure content acquisition from newsgroup (NNTP) servers by configuring and scheduling one or more News crawlers to access the desired directories and copy new or modified content into the application.

You can configure multiple News crawlers to access:

- documents located on multiple NNTP servers
- documents on a single NNTP server that have different collection requirements

Each unique crawler configuration defines a document collection.

You specify collection criteria for each collection by specifying common and crawler-specific configuration items. Specify the following common configuration items:

- the collection name, as described in ***Specifying Collections*** on page 14
- optional document filters, as described in ***Configuring Document Filters*** on page 17
- optional document attributes, as described in ***Configuring Document Attributes*** on page 17
- optional document supertitles, as described in ***Configuring Document Supertitles*** on page 18
- the document retrieval URL, as described in ***Specifying Full Document Retrieval URLs*** see "Specifying Document Retrieval URLs " on page 19

Specify the News crawler-specific parameters as described in ***Specifying*** see "Specifying News Crawler Parameters" on page 32News ***Crawler Parameters*** see "Specifying News Crawler Parameters" on page 32.

Specifying News Crawler Parameters

Specify the following News crawler-specifc configuration items in addition to the common configuration items:

Host

Specify the host name of the NNTP server.

Port

Specify the port name of the NNTP server.

Number of Initial Messages

Specify the maximum number of articles to process during a complete crawl. Any integer is valid. The default is to process all articles.

Retry Count

Specify the number of times that the application will try to reconnect if the connection is disrupted during content acquisition. Any integer is valid. The default is `0`, specifying no reconnection attempts.

Retry Sleep

Specify the length of the interval in seconds that the will wait between attempts to reconnect to the host if the connection is disrupted during content acquisition. Any integer is valid, and specifies a number of milliseconds.

Group

Specify one or more newsgroup names to collect content from.

Date Format

Specify any valid date formats that the application must process in addition to the standard packaged date formats. The standard date formats are:

```
<day-abbr>, <day-of-month> <name-of-month> <year>
<24hour>:<minutes>:<seconds> <timezone>
```

and

```
<day-of-month> <name-of-month> <year> <24hour>:<minutes>:<seconds>
<timezone>
```

where:

<day-abbr>	accepts the following weekday abbreviations; Mon, Tu, Wed, Thu, Fri, Sat, Sun
<day-of-month>	accepts integers `1-31`

<name-of-month>	accepts month names as full words, abbreviations, and numerals
<year>	accepts two- and four-character date designations
<24hour>:<minutes>:<seconds>	accepts two-character hour, minute, and second designations
<timezone>	accepts three-character time zone desgnations

Expiration Mode

Specify one of the following document expiration modes:

None	specifies that articles are never removed from the content store, even if they are removed from the server.
Lazy	only articles having ids lower than the lowest currently available article id on the server will be deleted from the content store.
Aggressive	specifies that the crawler will test that each article is still available on the NNTP server, and will mark any unavaliable articles for deletion from the content store.

Configuring File Systems

You configure content acquisition from file servers by configuring and scheduling one or more File crawlers to access the desired directories and copy new or modified content into the application.

You can configure multiple File crawlers to access:

- documents located on multiple file servers
- documents on a single file server that have different collection requirements

Each unique crawler configuration defines a document collection.

You specify content collection criteria for each collection by specifying common and crawler-specific configuration items. Specify the following common configuration items:

- the collection name, as described in ***Specifying Collections*** on page 14
- optional document filters, as described in ***Configuring Document Filters*** on page 17

- optional document attributes, as described in ***Configuring Document Attributes*** on page 17
- optional document supertitles, as described in ***Configuring Document Supertitles*** on page 18
- the document retrieval URL, as described in ***Specifying Full Document Retrieval URLs*** see "Specifying Document Retrieval URLs " on page 19
- the document type detector, as described in ***Specifying Document Type Detectors*** on page 20
- specify the starting point for the crawling process.

You can specify multiple starting points for a single crawler configuration. The crawler will access files and directories below each starting point based on its file and document filter configurations.

You specify starting points using the full pathname. Each starting point must be accessible from the application.

Configuring Content Acquisition from Databases

You access unstructured content within databases by configuring and scheduling one or more database crawlers to access the desired databases and copy new or modified content into the application. InQuira 6 provides a single content crawler type to access two types of database content:

- documents stored in binary format within a database, such that each record contains one document
- structured data, such as product catalogs, where each row of collected data is stored as a sentence in a single indexed document

You can configure multiple database crawlers to define various collections of database content. Each unique crawler configuration defines a collection, and each document can belong to only one collection. See ***Specifying Collections*** on page 14 for more information.

Note: See ***Acquiring Unstructured Content*** on page 13 for information on accessing unstructured content from servers, file systems, and other repositories.

You specify criteria for each collection by specifying common and crawler-specific configuration items. Specify the following common configuration items:

- the collection name, as described in ***Specifying Collections*** on page 14

- optional document filters, as described in ***Configuring Document Filters*** on page 17
- optional document attributes, as described in ***Configuring Document Attributes*** on page 17
- optional document supertitles, as described in ***Configuring Document Supertitles*** on page 18
- the document retrieval URL, as described in ***Specifying Full Document Retrieval URLs*** see "Specifying Document Retrieval URLs " on page 19

Specify the database crawler-specific parameters as described in ***Specifying Database Crawler Parameters*** on page 35.

Specifying Database Crawler Parameters

Specify the following Database crawler-specifc configuration items in addition to the common configuration items:

Data Source

Specify or define a datasource for the crawler.

Query

Specify an SQL query to retrieve relevant records. You can specify queries to retrieve structured data and documents stored within databases. For structured data, each record that you collect is stored as a sentence within the Content Store. For stored documents, each record corresponds to a document. You retrieve stored document content using an additional content query, as described below.

The query must select the following items in order:

- keys, if required, to access the content
- the string representation of the DocumentType (for example, HTML)
- the relevant content

Content Query

Specify an SQL query populate the rows extracted by the initial database query for stored document content. Use the Content Query Key Count to specify the number of key fields to bind to the preparedStatement.

Content Query Key Count

Specify the number of keys for the Content Query, if required.

Creating Custom Content Crawlers

You can create and configure custom content crawlers to integrate InQuira 6 with, and access data from, non-standard data sources. InQuira 6 provides flexible configuration for custom content crawlers based on name-value pairs.

You can create a custom crawler by:

- developing a class that extends the custom crawler configuration class, `com.inquira.content.custom.CustomCrawlerConfig`, as described in ***Extending the Custom Crawler Configuration Class*** on page 36
- developing a class that extends the content crawler class, `com.inquira.content.Crawler`, as described in ***Extending the Content Crawler Class*** on page 39
- configuring a collection to use the custom crawler, as described in ***Configuring a Collection to Use a Custom Crawler*** on page 43

Extending the Custom Crawler Configuration Class

To implement a custom content crawler, you must develop a new class that extends the custom crawler configuration class:

`com.inquira.content.custom.CustomCrawlerConfig`

The new custom crawler class must include a constructor that has public access, and must implement the following methods:

- ***The isModified Method*** on page 37
- ***The getCrawler Method*** on page 37

The new custom crawler class can override the following methods:

- ***The fetchExistingContent Method*** on page 37
- ***The init Method*** on page 38
- ***The buildDisplayURL Method*** on page 38

Custom Crawler Configuration Class Methods

The custom crawler class contains the following methods:

- ***The fetchExistingContent Method*** on page 37

The fetchExistingContent Method

Method: ***fetchExistingContent***

Syntax: `public boolean fetchExistingContent( )`

Description: This method specifies to retrieve all current documents from the Content Store and pass the resulting collection into the `findUpdatedContent` method.

Returns: the document collection from the Content Store

The isModified Method

Method: ***isModified***

Syntax: `public boolean isModified( Document doc1, Document doc2 )`

where:

doc1 is a document currently in the content store

doc2 is a newly disovered document from a crawler

Description: This method supports comparison of `doc1` and `doc2` by one of the comparison subclasses. The `CheckSumConfig` and `ModificationDateConfig` subclasses implement this method.

Returns: a value of `true` or `false`

The getCrawler Method

Method: getCrawler

Syntax: `public Crawler getCrawler( ) throws CrawlerException`

Description: This method returns a fully configured `Crawler` object that can crawl the custom datasource. Most implementations will return a new instance of the custom crawler class in this method.

Returns: a `Crawler` object

The init Method

Method: init

Syntax: `public void init( Properties configValues )`

where:

configValues is a list of name/value pairs

Description: This method stores the configuration name-value pairs in the instance variable named `configValues`. This method is called by the application when a custom crawler configuration object is created.

The buildDisplayURL Method

Method: buildDisplayURL

Syntax: `public String buildDisplayURL( Document doc ) throws CrawlerException`

where:

doc is a document

Description: This method returns the URL to be served by the User Interface to enable user access to the document identified by doc.

Returns: the URL as a string

Extending the Content Crawler Class

To implement a custom content crawler, you must develop a new class that extends the content crawler class:

`com.inquira.content.Crawler`

The implementing class must implement the following methods:

- ***The findUpdatedContent Method*** on page 39
- ***The getContent Method*** on page 40
- ***The connect Method*** on page 40
- ***The rundown Method*** on page 41
- ***The start Method*** on page 41
- ***The findComplete Method*** on page 41
- ***The getType Method*** on page 42

It can optionally override the `findUpdatedContent` method.

Content Crawler Class Methods

The content crawler class contains the following methods:

- The findUpdatedContent Method
- The getContent Method
- The connect Method
- The rundown Method
- The start Method
- The findComplete Method
- The getType Method
- The findUpdatedContent Method

The findUpdatedContent Method

Method: ***findUpdatedContent***

Syntax:

```
public Collection findUpdatedContent(
ExecutionContext cntx, CrawlerConfig cc, Collection
inCurrentContent, Collection inNewContent )
```

where:

cntx is the execution context

cc is an instance of the crawler configuration class

inCurrentContent is a list of documents from the Content Store

inNewContent is a list of documents generated by the crawl process

Description: This method compares the two collections of Document objects that are ordered by their external URL (the URL used to fetch the document). You can override this method if your custom crawler accesses a content store that records additions, modifications, and deletions. In this case the `fetchExistingContent` methods should return `false`.

Returns: a collection of document objects

The getContent Method

Method: ***getContent***

Syntax:

```
public byte[] getContent( CrawlerConfig cc,
Document doc ) throws CrawlerException
```

where:

cc is an instance of the crawler configuration class

doc is a document

Description: This method retrieves the raw content for the specified Document `doc`.

Returns: the raw document content in bytes

The connect Method

Method: connect

Syntax: `public void connect( CrawlerConfig cc ) throws CrawlerException`

where:

cc is an instance of the crawler configuration class

Description: This method is called before any other operation on the crawler to allow it to establish a connection to the underlying content repository.

The rundown Method

Method: rundown

Syntax: `public void rundown( )`

Description: This method is called after the Content Store predicts that no future operations are expected on this content source.

The start Method

Method: start

Syntax: `public void start( )`

Description: This method is called to signal a new crawl to the underlying content repository. You can use this method if the target content repository is stateful or needs to be informed that a new crawl has started for some reason.

The findComplete Method

Method: findComplete

Syntax: `public boolean findComplete( )`

Description: The content acquisition framework allows for partial crawls, thus limiting the amount of data/documents to process. The framework will call this method in a loop until it returns `true`. For simple crawlers this should always return `true`.

Returns: a value of `true` or `false`

The getType Method

Method: ***getType***

Syntax: `public ContentSourceType getType`

Description: This method returns one of the predefined `ContentSourceTypes` in `Document`.

Managing Custom Crawler States

You can include functionality to manage the state of a custom crawler using the `CrawlerState` class.

The CrawlerState Class

The `CrawlerState` class can hold any Serializable object that represents the state of the custom crawler. If the state of the crawler has to be persisted, the state object must be accessed in the `CrawlerState` object that is passed into the `Crawler.findCrawler` method using the `public set( Serializable )` method to store and `public Serializable get( )` to retrieve the crawler-specific state information.

Custom Crawler Configuration Class Sample Code

The following sample shows the code for the custom crawler configuration class, `CustomCrawlerConfig`.

```
package com.inquira.content.custom;

import java.util.*;

import com.inquira.infra.*;
import com.inquira.config.*;
import com.inquira.content.*;

public abstract class CustomCrawlerConfig
    extends CrawlerConfig
{
    private static final String __ident = "$Revision: 1.2 $";


    protected Properties configValues;
```

```
    public CustomCrawlerConfig( )
    {
    }

    public void init( Properties configValues )
    {
        this.configValues = new Properties( );
        if( configValues != null ) {
            this.configValues.putAll( configValues );
        }
    }

    public String buildDisplayURL( Document doc )
        throws CrawlerException
    {
        return doc.getDisplayURL( );
    }

    public abstract boolean isModifiedDocument( Document currentDocument,
                                                Document newDocument );

    public abstract Crawler getCrawler( )
        throws CrawlerException;

}
```

Configuring a Collection to Use a Custom Crawler

You configure a collection to use a custom crawler by specifying:

- the name of custom crawler configuration class, as defined in ***Extending the Custom Crawler Configuration Class*** on page 36, and as described in ***Configuring Content Crawlers*** on page 13
- the custom crawler specific configuration in name-value pair format:
    - a optional list of Document Filter(s), as described in ***Configuring Document Filters*** on page 17
    - an optional list of Document Attribute Selectors, as described in ***Configuring Document Attributes*** on page 17
    - an optional list of Supertitle Selectors, as described in ***Configuring Document Supertitles*** on page 18
    - an optional Document Type Detector, as described in ***Specifying Document Type Detectors*** on page 20

Sample Collection Configuration

The following sample shows a collection configured for a custom content crawler as represented in the application configuration file.

```
<type name="CustomCrawlerConfig" printname="CustomCrawlerConfig"
administrable="true"
controller="com.inquira.content.custom.CustomCrawlerConfigController">

    <entry name="class" printname="class" type="className" />

    <entry name="config" printname="config" type="Properties"/>

    <entry name="unstructuredAttribute" type="boolean" default="true"/>

    <list name="documentFilter"
controller="com.inquira.content.DocumentFilterController"
refkey="choices.simpleDocumentFilter" />

    <list name="documentAttributeSelector" type="DocumentAttributeSelector"
/>

    <list name="documentSupertitleSelector"
type="DocumentSupertitleSelector"/>

    <entry name="documentTypeDetector"
refkey="choices.magicDocumentTypeDetector" type="DocumentTypeDetector"/>

</type>
```

Accessing Structured Content

You can configure InQuira 6 to access structured information within databases or in other structured formats, such as Extensible Markup Language (XML). InQuira 6 accesses structured data by connecting directly to a configured data source. It does not collect and store structured data for content processing and language analysis, but instead queries the data source directly during request processing.

Note: You can also configure InQuira 6 database crawlers to collect unstructured data, such as documents stored within databases or free text within database fields, thereby making that content available for content processing and language analysis. See ***Associating a Data Source with a Configured Database Crawler Collection*** on page 49 for more information.

InQuira 6 accesses structured information by:

- translating user requests into structured queries
- directly querying the data source
- filtering the query results, if required
- formatting and presenting the results

Translating User Requests into Queries

The Rules Engine uses Rules within the Dictionary to translate user requests into structured queries. You can configure Rules to produce SQL queries that match the intent of users' natural language questions using the structured query method, as described in *InQuira 6 Language Guide*.

Querying the Data Source

Structured query Rules use pre-defined internal schemas to produce SQL queries. The structure of the SQL queries are determined by the pre-defined schema.

To enable InQuira 6 to generate SQL queries to locate and retrieve the appropriate data, you need to map the structure of your data source to the structure of the pre-defined schema, as described in ***Configuring Relational Database Sources*** see "Configuring Structured Data Sources" on page 46.

Restricting Query Results

You can restrict the results of structured queries by including additional IML statements within Rules. The Rules Engine processes IML associated with structured queries by:

- integrating it into the initial query

or

- executing the query, then using the results set as the input for IML processing

You can configure the use of these alternate processing methods to optimize structured query processing for your application by specifying the Maximum IML Results in Query global parameter, as described in ***Setting the Maximum Number of Structured Answers*** on page 50.

Displaying Structured Results

The User Interface displays structured results in a tabular format that includes a table heading. You specify the table heading in the Title of Results Rule parameter as described in the *InQuira 6 Language Guide*.

The User Interface also displays a natural language paraphrase of the SQL query. The User Interface uses the following configurable parameters to construct the paraphrase:

- the table description
- the table key
- the field description

as described in ***Viewing Schema Details*** on page 47.

Configuring Database Connections for Structured Content

You can configure multiple structured data sources, including relational database tables and XML data repositories. You configure structured information retrieval by:

- configuring the desired data sources, as described in ***Configuring Relational Database Sources*** see "Configuring Structured Data Sources" on page 46 and ***Configuring XML Data Sources*** on page 48
- setting global parameters for structured information retrieval, as described in ***Specifying Global Structured Data Retrieval Parameters*** on page 49

Configuring Structured Data Sources

The database retriever receives SQL from the Rules Engine and uses the configured internal schema to transform the query into an appropriate query for the configured database. The transformation includes removing unavailable fields and substituting alternate fields, if configured to do so. It executes the query, processes any associated IML as described in ***Configuring IML Processing for Structured Queries*** on page 50, and returns the results.

You configure a relational database by:

- creating a custom view of your data source, as described in ***Creating a Database View*** on page 46
- configuring the appropriate application schema to connect to the custom data source, as described in ***Collection*** see "Associating a Data Source with a Configured Database Crawler Collection" on page 49 ***Viewing Configured Schema*** see "Viewing Configured Schema" on page 47
- specifying an optional collection for unstructured content processing, as described in ***Associating a Data Source with a Configured Database Crawler*** see "Associating a Data Source with a Configured Database Crawler Collection" on page 49

Creating a Database View

To configure InQuira 6 to retrieve answers from your data, you need to create a view that is compatible with the structure of the appropriate internal schema. For information on available internal schemas, contact your Technical Account Manager.

Viewing Configured Schema

InQuira 6 is installed with pre-defined schemas. You can view the available schemas, and their defined keys and fields using the Configuration and Administration Interface.

To view the available schemas:

- select Structured Services from the Configuration and Administration Interface

The Structured Services page lists the installed schemas.

Viewing Schema Details

To view details for a schema:

- select Edit on the Structured Services page

The Editing:Structured Services page displays the schema names as hypertext links

- select the desired table name from the list

The Editing:Structured Services page displays the following fields for the selected table:

Field	*Description*
Description	A natural language representation of this table that will be used in the paraphrase. For example, `real-time stocks`
Data Source	The location of the data.
Collection	The crawler configuration defined for acquiring unstructured data used for hybrid retrieval.
Key	The field in this schema that is the key (unique identifier). This is used when creating the paraphrase. For example, `stocksymbol`.
Field	A list of the fields in the schema. ***Schema Field Definitions*** on page 47 describes the parameters that define fields.

Schema Field Definitions

Each field in the schema is defined by the following parameters:

Field	*Description*
Description	A natural language representation used in paraphrase. For example, `book value in dollars`.

Exists	Whether or not it exists in this particular database. Possible values are `true` (default) and `false`.
Groupable	Not currently implemented.
Alternate	Specifies the field to ues if this field does not exist in the data source.

Connecting to the Data Source

To configure InQuira 6 to retrieve answers from a data source, specify the data source connection parameters, as described in ***Specifying Data Source Connection Parameters*** on page 5.

Specifying Data Source Connection Parameters

Specify the following parameters to connect the InQuira 6 application to a data source.

URL

Enter the connection URL for the JDBC connection. The value of the connection URL depends on your RDBMS and JDBC driver.

Driver Class Name

Enter the fully qualified JDBC driver name.

Exception Handler

The exception handler checks for known error conditions, such as unique key violations. You can configure one or more of the appropriate pre-defined exception handlers for re-use in multiple application configurations. InQuira 6 supplies the following pre-defined exception handlers:

- Oracle 9i
- Microsoft SQLServer 2000
- MySQL 3

Connection Properties

Specify any connection properties, such as user name and password, as required.

Configuring XML Data Sources

The XML retriever receives SQL from the Rules Engine and uses a configured script to translate the SQL into XML. It sends the XML query to a configured XML transport interface that communicates with the XML data source. It uses an additional script to translate the XML results into a tabular results set for display.

Query Set DTD on page 53 describes the DTD for query scripts. ***Result Set DTD*** on page 54 describes the DTD for the results set.

You configure an XML data source by:

- creating and configuring query and results scripts
- implementing the XML transport interface
- specifying an optional collection for unstructured content processing, as described in ***Configuring IML Processing for Structured Queries*** on page 50

Associating a Data Source with a Configured Database Crawler Collection

When you configure a database crawler to collect unstructured data stored within a database, you need to:

- configure a data source for the collection, as described in ***Specifying Data Source Connection Parameters*** on page 5
- associate the database crawler collection with the schema defined for the data source

You associate a database collection with a data source using the Structured Services page of the Configuration and Administration Interface. To associate a collection with a schema:

- select an existing schema, or select Add New Item

The Editing:Structured Service > Table page displays.

- select an existing collection, or create a new collection as described in ***Configuring Content Acquisition from Databases*** on page 34

Specifying Global Structured Data Retrieval Parameters

You specify the following global parameters for structured information retrieval:

- the maximum number of answers that will be returned in response to a structured query, as described in ***Setting the Maximum Number of Structured Answers*** on page 50
- the maximum number of IML query results to integrate into a structured query, as described in ***Configuring IML Processing for Structured Queries*** on page 50

- the XML translation scripts to use for retrieving structured information from XML data sources, as described in ***Specifying XML Translation Scripts*** see "XML Translation Scripts" on page 51
- whether the configured data sources support sub-queries, as described in ***Specifying Sub-Query Processing*** on page 51
- the XML transport interface, as described in ***Specifying the XML Transport Interface*** on page 51

Setting the Maximum Number of Structured Answers

You specify the maximum number of answers that structured information retrieval will return for a single query. The default value is 500.

The application will retrieve the up to the specified number of results for every query. If a user requests quantities in excess of the specified value, the application will retrieve only the maximum number of results.

You specify the maximum number of answers on the Structured Service page of the Configuration and Administration Interface. To specify the maximum number of answers:

- specify a value maximum answers value. Any integer is valid. The default value is `500`.
- select OK to save the specified value in your configuration

Configuring IML Processing for Structured Queries

The Rules that generate SQL queries can also contain IML to restrict the results of the query. When the IML query is executed, the application:

- integrates the IML statements into the structured query

or

- processes the IML using the results of the structured query as input

A large number of IML results can increase the size of the query such that the database will reject it. The Maximum IML Results in Query specifies the number of results over which the application will post-process the IML results rather than integrate them into the structured query.

You specify the maximum IML results in query on the Structured Service page of the Configuration and Administration Interface. To specify the maximum IML results in query:

- specify a maximum IML results value. Any integer is valid. The default value is `500`

- select OK to save the specified value in your configuration

XML Translation Scripts

To configure an XML data source, you need to create scripts to translate:

- InQuira 6 SQL queries to client XML queries
- XML results to InQuira 6 XML results

Query Set DTD on page 53 and ***Result Set DTD*** on page 54 provide document type definitions (DTDs) for the query and results scripts.

You specify query and results scripts for structured retrieval on the Advanced Structured Service page of the Configuration and Administration Interface. To specify the query and results scripts for structured retrieval:

- select Advanced from the dropdown menu on the Edit: Structured Service page
- enter the URLs for the query and results scripts
- select OK to save the specified value in your configuration

Specifying Sub-Query Processing

The SQL queries produced by the application can contain subqueries; however, some databases do not support subqueries. This option specifies to execute subqueries as separate queries.

You specify subquery processing on the Advanced Structured Service page of the Configuration and Administration Interface. To specify subquery processing:

- select Advanced from the dropdown menu on the Edit: Structured Service page
- select the On radio button to specify standard subquery processing
- select the Off radio button to specify that subqueries will be executed as separate queries
- select OK to save the specified value in your configuration

Specifying the XML Transport Interface

Specify a class that implements the XMLTransport interface. This class receives the client's xml query (as a string) and returns the results in the client's xml representation (as a string).

Creating Custom Structured Data Source

Connectors

You can create and configure connectors for additional structured data sources. To create a custom connector, you must:

- implement the `StructuredDataRetriever` interface, as described in ***The Structured Data Retriever Interface*** on page 52
- (optional) extend the `StructuredDataAdapter` class which contains helpful code for getting IML results.

You configure the custom connector using the Advanced Structured Service page of the Configuration and Administration Interface. To configure the custom connector:

- select Add New Item under the Connection Type field

The Editing: Structured Service > Connection Type page displays.

- enter the class name for the custom connector in the Connection field, and select Add
- select OK to save the specified value in your configuration

The Structured Data Retriever Interface

The structured data retriever interface has the following method:

```
public StructuredResult evaluate(Schema schemas,
                                 String schemaName,
                                 String query,
                                 String iml,
                                 Datasource data_source,
                                 String index,
                                 Map options
                                 ) throws StructuredException;
```

where:

Argument	Description
schemas	are the schemas that were configured for the various types of sources (stocks, mutual funds, etc). This is a datastructure that allows easy access.
schemaName	is the name of the schema to use when querying the `schemas` data structure
query	is the structured query to execute

iml	is the restricting IML query that will be run against the unstructured content retrieved from the datasource
data_source	is the datasource to which to send the structured query
index	is the name of the collection that contains the unstructured data to be queried by the IML argument
options	is a map of the configuration for structured search

Query Set DTD

The dtd for the query:

<!ELEMENT QUERY (FIELD*, FCN*, CONST*, SOURCES, QUALIFICATION?, LIMIT?, ORDER?) >

<!ATTLIST QUERY op(SELECT) >

<!ELEMENT SOURCES (TABLE+) >

<!ELEMENT TABLE (#PCDATA) >

<!ELEMENT LIMIT (START?, QUERY?, COUNT) >

<!ELEMENT ORDER_BY (ORDER+) >

<!ELEMENT ORDER (FIELD) >

<!ATTLIST ORDER dir NMTOKEN #REQUIRED >

<!ELEMENT CONST (#PCDATA) >

<!ELEMENT FCN (ORDER+) >

<!ATTLIST FCN name #REQUIRED >

<!ATTLIST FCN printname #IMLPLIED >

<!ELEMENT FIELD (ORDER+) >

<!ATTLIST FIELD name #REQUIRED >

<!ATTLIST FIELD printname #IMLPLIED >

<!ELEMENT QUALIFICATION
(AND|OR|LIKE|IS_NOT|NE|LT|LTE|GT|GTE|EQ) >

<!ELEMENT AND (AND|OR|LIKE|IS_NOT|IN|NE|LT|LTE|GT|GTE|EQ)+>

<!ELEMENT OR (AND|OR|LIKE|IS_NOT|NE|LT|LTE|GT|GTE|EQ)+>

<!ELEMENT NE ((CONST|FIELD|FUNC),(CONST|FIELD|FUNC))>

<!ELEMENT EQ ((CONST|FIELD|FUNC),(CONST|FIELD|FUNC))>

<!ELEMENT LT ((CONST|FIELD|FUNC),(CONST|FIELD|FUNC))>
<!ELEMENT LTE ((CONST|FIELD|FUNC),(CONST|FIELD|FUNC))>
<!ELEMENT GTE ((CONST|FIELD|FUNC),(CONST|FIELD|FUNC))>
<!ELEMENT LIKE ((CONST|FIELD|FUNC),(CONST|FIELD|FUNC))>
<!ELEMENT IS_NOT ((CONST|FIELD|FUNC), NULL))>
<!ELEMENT NULL>

Result Set DTD

The dtd for the result set is:

<!ELEMENT const (#PCDATA) >
<!ELEMENT field EMPTY >
<!ATTLIST field name NMTOKEN #REQUIRED >
<!ELEMENT headers (field) >
<!ELEMENT resultset (headers, row+) >
<!ELEMENT row (const) >

Chapter 6

Translating Content to the InQuira Format

The Preprocessor translates the documents collected by the content acquisition process from their native formats to a standard, simplified XML format. The standard XML format is called InQuira Input XML, and is defined by a the document type definition (DTD) described in ***The InQuira Input XML DTD*** on page 84. Supported document formats are described in ***Supported Data Formats and Converters*** on page 57.

The Preprocessor contains several component programs, called converters. Each converter is designed and configured to analyze and translate a specific document format.

The Preprocessor is configured at installation to:

- locate content to be processed
- determine document types
- invoke the appropriate converters to translate all documents to InQuira Input XML
- store the InQuira Input XML documents in the Content Store

Note: You must have a configured and populated Content Store for preprocessing to occur. See ***Configuring Content Acquisition*** on page 11.

Preprocessing Overview

During content processing, the Preprocessor:

- obtains a list of available documents and their status from the Content Store
- determines the files to be processed, and their document types
- translates the files to InQuira Input XML
- stores the InQuira Input XML documents in the Content Store

The InQuira Input XML format is defined by the InQuira Input XML document type definition (DTD), which is described in detail in ***The InQuira Input XML DTD*** on page 84.

Translation Accuracy

In its default configuration, the Preprocessor produces InQuira Input XML-compliant documents that provide semantic information used in processing requests. The Preprocessor translates all input data without loss, and provides excellent request processing and information retrieval accuracy. However, you may encounter circumstances in which you need to optimize content processing or tailor it to the specific requirements of your organization. See ***Customizing the Preprocessor*** on page 60 for information on customizing the Preprocessor and its components.

Translating Document Structure

The Preprocessor converters are configured to retain semantic information encoded in the document structure, such as headings, paragraphs, and lists, when translating documents. The converters translate the semantic elements located within the original documents to analagous elements defined in the InQuira Input XML DTD.

Access to Document Structure During Request Processing

During request processing, InQuira Match Language (IQIML) functions within the Rules can access InQuira Input XML semantic structures to optimize request processing. The following IML functions can access InQuira Input XML entities:

TITLE

This function specifies to match within document titles.

SUBTITLE

This function specifies to match within document subtitles, including section headings and other named sections of documents.

REFERENCE

This function specifies to match within documents that are linked to by specified terms within link text.

See *InQuira 6 Language Guide* for more information on specifying IML functions.

Processing Non-ASCII Characters

ThePreprocessorautomatically strips out unicode characters that it encounters within the application content. The XML::Writer Perl module cannot process unicode characters and will fail if they are present.

Important: Converting Unicode characters to numeric or named entities is not sufficient, since the XML::Parser translates them back to character format.

The Preprocessortranslates Latin 1 – ISO characters (character code 131 – 256) to numeric entity names.

Supported Data Formats and Converters

The Preprocessor contains converters to translate various document formats to the standard InQuira Input XML format. Each document format is associated with a specific converter module.

Converter	*Document Type*	*File Extensions*
General Text Document	HTML, XML, Microsoft Word, Newsgroups, ASCII text, unstructured text within databases	.htm, .html, .xml, .doc, .txt
PDF	Portable Document Format (PDF)	.pdf
Excel	Microsoft Excel	.xls
PowerPoint	Microsoft PowerPoint	.ppt

Note: The Preprocessor uses its converters as libraries; you cannot operate the converters independently of the Preprocessor.

For information on processing additional document formats, see ***Creating Custom Converters*** on page 73.

Processing HTML Documents

The Preprocessor uses the general text document converter to translate hypertext mark-up language (HTML) files to InQuira Input XML.

It translates documents by translating the HTML tag set into corresponding InQuira Input XML elements, using script directives and tag translations specified in its configuration file. You can tailor translation specifically for your site's content, if necessary, using the process described in ***Customizing HTML Translation*** on page 60.

Processing XML Documents

The Preprocessor uses the general text document converter to translate extensible mark-up language (XML) files to InQuira Input XML.

It translates documents by translating the source XML tags into corresponding InQuira Input XML elements, using script directives and tag translations specified in its configuration file. You can tailor translation specifically for your site's content, if necessary, using the process described in ***Customizing XML Translation*** on page 68.

Processing Microsoft Word Documents

The Preprocessor uses the general text document converter to translate Microsoft Word documents to InQuira Input XML.

The general text document converter uses the wvWare application and wvHtml.xml entity mapping file to translate Microsoft Word documents to HTML. The converter then uses the HTML module to translate HTML to InQuira Input XML.

You can customize Microsoft Word document translation using the HTML translation customization process described in ***Customizing HTML Translation*** on page 60.

Note: Changes to the HTML translation configuration will also have an effect on Microsoft Word translation.

Processing PDF Documents

The Preprocessor uses the PDF converter to translate Portable Document Format (PDF) files to InQuira Input XML. The PDF converter contains heuristics for determining the following semantic information from the text provided in the PDF file:

- titles and sections are determined by the PDF title tag; they are also inferred from internal bookmark information, if available
- paragraphs are determined by sentence proximity
- additional heuristics distinguish other semantic features, such as tables and multi-column text

Important: The Preprocessor cannot translate encrypted PDF files, because encrypted files cannot be copied. You must use non-encrypted PDF files for InQuira 6 content processing.

Processing ASCII Text Documents

The Preprocessor uses the general text document converter to translate ASCII text files, Newsgroup messages, and free text stored within database records to InQuira Input XML.

Processing Microsoft Excel Documents

The Preprocessor uses the Excel converter to translate Microsoft Excel documents to InQuira Input XML. The Excel converter uses the JExcel application to convert Excel workbook elements to InQuira Input XML elements.

The converter translates each worksheet as a section. Within each section, text is processed as follows:

- the name of the worksheet is tagged as the section title
- each row in the sheet, as defined by the Excel application, is tagged as a sentence

Processing Microsoft PowerPoint Documents

The Preprocessor uses the PowerPoint converter to translate Microsoft PowerPoint documents to InQuira Input XML. The PowerPoint converter operates on a dedicated NT server, and uses the Microsoft PowerPoint application to perform the translations. For more information on using the Microsoft PowerPoint Converter, see ***Using the PowerPoint Converter*** on page 70.

The PowerPoint converter translates every slide as section. Within each section, text is processed as follows:

- the text box closest to the top edge of the slide is tagged as the section title
- the slide number is tagged as the section anchor
- text associated with a shape (as defined by the PowerPoint application) is tagged as a single paragraph
- tables are translated in the same way as Microsoft Word tables
- grouped shapes are ungrouped until all associated text can be extracted
- text in the notes section is tagged as a paragraph

Using the Preprocessor

You use the Preprocessor by scheduling preprocessing tasks, as described in ***Scheduling InQuira Processes*** see "Scheduling InQuira 6 Processes" on page 115. The Preprocessor is

configured at installation to locate the content to be processed, and write its output to the Content Store.

You can coordinate preprocessing with the other content processing tasks, or operate the Preprocessor independently of the content processing tasks during application development and testing.

You can modify the converters to accomodate specific structural and semantic requirements of your content, as described in ***Customizing the Preprocessor*** on page 60. You can also create and register custom converters to:

- preprocess document types for which InQuira 6 does not provide a converter
- override the InQuira 6 converter for specific document types

See ***Creating Custom Converters*** on page 73 for information on creating and registering custom converters.

Customizing the Preprocessor

You can tailor preprocessing to optimize language processing and request processing for various document types by modifying the corresponding converter configuration files for the following document formats:

- HTML (see ***Customizing HTML Translation*** on page 60)
- XML (see ***Customizing HTML Translation*** on page 60)
- Microsoft Word (see ***Processing Microsoft Word Documents*** on page 58)

You can also create and register custom converters to process addtional document types or to override the standard converters, as described in ***Creating Custom Converters*** on page 73, if required.

Customizing HTML Translation

You customize HTML translation by specifying processing directives and tag translations to optimize content processing and information retrieval. You specify tag translations and processing directives in the HTML converter configuration file, as described in ***HTML Conversion Module Configuration File*** see "HTML Converter Configuration File" on page 61 .

Important: The HTML converter is also used as a step in the Microsoft Word translation processes. Changes to the the HTML translation configuration will also have an effect on Microsoft Word translation.

HTML Converter Configuration File

The HTML converter configuration file is called `html.config`. It is located in the InQuira 6 `src/prep` subdirectory.

The configuration file contains the instructions for translating tags and other semantically significant information supplied within HTML documents.

The configuration file contains a set of translation commands that are executed by the converter. The translation commands are tab delimited, and contain spaces for tag attributes. For example:

```
p class="headmain" <tab> title
```

HTML Converter Configuration Requirements

The configuration file has the following requirements:

- processing of tag and attribute names is case-insensitive
- processing of attribute values is case-sensitive
- the HTML conversion module ignores any tag not specified in the configuration file. The text associated with the tag will be treated as paragraph text.
- space between words will be preserved; no word splitting or concatenation will occur

Tag Conversion Command Types

There are two types of commands that control HTML translation:

Note: In the following examples, arrows (->) represent tabs.

Processing directives

Processing directives specify an action to perform when the specified element is encountered. The specified action can be a translation, or the addition of new information. For example:

```
h1 -> !section1
```

Tag translations

Tag translations specify to translate instances of one tag or entity to instances of another. For example:

```
h1 -> title
```

You can specify simple and complex tag translations, as described in ***Specifying Tag Translations*** on page 62.

Tag Conversion Command Forms

You can specify both simple and complex translations and directives. Examples of simple and complex translations and directives are shown in the following table:

	Simple	*Complex*
Translation	`h1 -> title`	`dl|dd|dt -> p`
Directive	`h1 -> !section1`	`dl|dt -> !punct`

Specifying Tag Translations

The default settings in the HTML translation configuration file associate a set of HTML tags with corresponding XML tags. To override the default associations, specify each HTML tag and the corresponding XML tag that you want to translate to. For example:

```
<htmltag> -> <xmltag>
```

The default translation for HTML paragraph tags to XML tags is shown as:

```
<p> -> <p>
```

Important: The converter ignores any tag not specified in the configuration file. The text associated with the tag will be treated as paragraph text.

Specifying Simple Tag Translations

Use simple translations to translate:

Simple tags	`h1 -> title`
Named entities	`&quot -> "`
Attribute translations	`p class="h1" -> title` `p class -> title`

Note: If you define both an attribute translation such as `p class="h1" -> title` and a default translation, such as `p -> p`, the attribute translation will override the simple tag translation.

Specifying Complex Tag Translation

Use complex translations to translate larger tag structure patterns. An example of a complex translation is:

```
dl|dd|dt -> p
```

This specifies that a <dt> tag within both a <dl> tag and a <dd> tag:

```
<dl> …
<dd> …
<dt> …
</dt>
</dd>
</dl>
```

will be converted to <p>.

The resulting tag structure is:

```
<dl> …
<dd> …
<p> …
</p>
</dd>
</dl>
```

You can also use complex attribute translations to translate larger structures involving attributes and tags:

```
p class="list"|dd|dl -> p
```

or

```
p class="list"|dd|dl -> !section1
```

Simple Script Directives

Use simple directives to treat certain tag sets:

```
h1 -> !startsection1
```

In this example, you may want to translate the tag to a title but also create a section/subsection:

```
h1 -> title
h1 -> !startsection1
```

Translation Directives

Directive	Description
!keep	Keeps the tag in its current format with all attributes. To keep a tag without its attributes, use a simple translation: p -> p will do this for all tags without more specific attribute definitions. See ***Specifying Simple Tag Translations*** on page 62.
!hide	Hides the tag and any text associated with the tag. `p class="navbar" -> !hide` will exclude `<p class="navbar">HOME | OFFICE | BAR </p>` from the XML output.

!ignore	Ignores a tag. This is the default behavior when tag is not specified in configuration file. This is useful for special treatment of tags with attributes:
	`font -> !ignore`
	Otherwise, `p -> p` would save this tag.
	`p class="superfluous"  !ignore`

!listtitle	Translates the first tag in a list or table as the title for that section. All following tags of that type are ignored (`!ignore`).
	`table|tr|td -> !listtitle`
	`table -> !section2`
	The following example will translate a table to a section (of level 2), and will translate the first `<tr><td>...</td></tr>` tag set as the title for that section.
	Note: The title will be the only title within the 'table' tag. If you specify:
	`td -> !listtitle`
	then the title will be the only title within the `tr` tag set.

!startsection[1-6]	Begins a section immediately before the start of the tag. The number indicates the level of the subsection.
	`h1 -> !startsection1`
	`h2 -> !startsection2`
	where `h2` starts a subsection of the `h1` section.
	The section ends when a matching `!endsection[1-6]` is encountered, or when the script encounters a directive of equal or greater standing . In the above example another, `h1` tag would end the first section.

!endsection[1-6]	Ends a section immediately before the start of the tag. This is the complement to `!startsection[1-6]`. `Table -> !endsection5`

!section[1-6]	Translates a tag to a section tag. This directive works with `!startsection[1-6]` and `!endsection[1-6]` directives to ensure well-formed XML. `para -> !section[4]`

!emptytag	Makes the current tag an empty tag (`<tag/>`), but keeps the tagged text.
	`a name -> !emptytag`
	`<a name="name"> .. </a> -> <a name="name" />`
	If the tag is already empty, you can use the `!keep` directive as follows:
	`Tag -> !keep`

!deletetext	Keeps a tag (makes it an empty tag) but deletes the tagged text.
	`meta -> !deletetext`
	`<meta> text to delete </meta>  <meta />`

!doctitle	Makes the particular tag the title for the document.

!space	Adds a space after the tag. This is helpful when tags are used to delimit words.
	`font -> !space`
	`<font type="big">HELP</font>ME  HELP ME`
	The font tag is processed by the default `!ignore` directive.

!punct	Creates a sentence break after a tag. This is useful for breaking up tables and unpunctuated lists. `Td -> !punct`

Specifying Complex Script Directives

You can use complex script directives to translate larger tag sets.

`dl|dd|dl -> !section1`

This is similar to complex tag translation in that a larger tag set is being referenced for the translation.

Conflicts Between Section Directives

The three section directives (`!startsection[1-6]`, `!endsection[1-6]`, `!section[1-6]`) resolve conflicts in the following manner:

- `!section[1-6]` directives have precedence over `!endsection[1-6]` tags.

- If the script encounters a `!startsection[1-6]` or `!endsection[1-6]` during parsing of `<tag> -> !section[1-6]` where `<tag>`'s section is of less or equal standing, then the script will demote the `!startsection` to a lower standing and defer the `!endsection` directives until the completion of `<tag>`'s translation.
- Startsection tags are demoted rather than deferred because startsection directives are associated with title translations. If a title tag is created in the middle of a section or document without first opening a section tag, ill-formed XML will result.

```
config file: table -> !section2
td -> !startsection1
```

- The script starts the `td section1` as `section3` standing.
- `!section[1-6]` directives demote `!section[1-6]` directives of equal or greater standing within their tag scope. The overridden directives are forced into a lower standing.

```
config file: table -> !section2
td -> !section2
```

- The script translates the `table` tag into a level 2 subsection, and translates the `td` tag into a level 3 subsection.
- Sections of lesser or equal standing are closed by new section directives.

```
config file: table -> !startsection2
td -> !startsection2
```

- The script closes the section that started at the table tag and begins another level 2 section.
- If the script encounters an `!endsection[1-6]` where there are no open sections at that level, it will ignore the directive.
- If the script completes parsing, and there are open sections, it will close all sections.

Conflicts Between Other Directives

All other directives and translations override each other based on inherent precedence properties or on the order of the configuration file. The last declaration processed overrides all previously processed declarations.

```
config file:  p -> !section1
p -> p
```

The converter will translate all p tags to p.

```
configuration file: p -> p
p class="alpha" -> !section1
```

The converter translates all p tags to p except for p tags with class attribute equal to alpha; these tags are translated to level 1 section tags.

Using Directives in Conjunction with Translations or Other Directives

You can use the following directives in conjunction with other directives and translations:

- `!startsection[1-6]`
- `!endsection[1-6]`
- `!space`
- `!punct`

For example, you could configure the `h1` tag in the following manner:

`h1 -> title`	Tags the h1 text with title tags.
`h1 -> !startsection1`	Starts a section prior to the title the tag.

Example Translations and Directives

Purpose	*Example*
Translate titles and set sectioning	`h1 -> title` `h1 -> !startsection1`

Translate other forms of heading tags	`p class="head1" -> title`
	`p class="head1" -> !startsection1`
	`p class="h1" -> title`
	`p class="h1" -> !startsection1`

Translate these tags	Pre -> data

Tables	`Table -> !section5`
	(Make this as low a section level as possible)
	`table|tr|table -> !section6`
	`td -> !punct`

Lists	`ul -> !section5`
	`ul|li|ul -> !section6`
	`dl|dt|dd|dl -> !ignore`
	`ol|li -> p`

Keep these tags	`p -> !keep`
Make empty tags	`a name -> !emptytag`
Translate named entities	`  -> !space` `  -> !space`
Hide these tags	`p class="navbar" -> !hide`
Specify document title	`p class="headmain" -> !doctitle`

Using the Document Title Directive

You define document titles using the !doctitle directive. The !doctitle directive replaces the HTML file's title with the specified tag. If a tag is not specified, or if you specify a tag that does not exist, then the Preprocessor uses the HTML file's title.

The Preprocessor performs the !doctitle directive prior to translation. The HTML conversion module cannot define the title in an event-driven manner. The document title is the first title (other than supertitle) and occurs prior to all sentence and paragraph text. The associated tag may not be the first occurring tag.

Customizing XML Translation

You customize XML translation by specifying processing directives and tag translations to optimize content processing and information retrieval. You specify tag translations and processing directives in the XML converter configuration file, as described in ***XML Converter Configuration File*** on page 68.

XML Converter Configuration File

The XML converter configuration `xml.config` is located in the `src/prep` subdirectory.

The configuration file contains the instructions for translating tags and other semantically significant information contained within XML documents. It contains a set of translation commands that are executed by the converter. The translation commands for customizing XML translation are identical to those described in ***Customizing HTML Translation*** on page 60.

Customizing PowerPoint Translation

Important: You must have the the PowerPoint converter installed and configured to preprocess Microsoft PowerPoint documents. See ***Using the PowerPoint Converter*** on page 70 for information.

The PowerPoint converter uses a configuration file. The default configuration file is shipped with the product. The default configuration file path name is `c:\_listener.xml`.

The first time you start the converter it uses the information in the configuration file to create registry entries for the configuration parameters. You can configure the converter by directly editing the registry on the dedicated NT server. The parameters set by the configuration file are:

Configuration file name

Specifies the configuration file. The default is `c:\_listener.xml`.

Log file name

Specifies the log file for the converter. The default is `c:\_listener.log`. The converter logs all messages. You can specify a separate log file by specifying a log file name as part of its communication to the converter. Only messages corresponding to that client will be logged to that file.

Port

Specifies the port on which the converter listens for client connections. The default value is `8123`.

Editing PowerPoint Converter Configuration Parameters

You can configure the converter by directly editing the registry on the dedicated NT server. To edit the registry:

- stop the master program
- open the regedit window
- locate the `HKEY_LOCAL_MACHINE/SOFTWARE/InQuira` statement
- edit the desired values
- restart the master program

Using the PowerPoint Converter

The PowerPoint converter operates on a dedicated NT server, and uses the Microsoft PowerPoint application to perform the translations.

To use the PowerPoint converter:

- install the PowerPoint converter
- specify the PowerPoint converter in the Preprocessor configuration
- specify PowerPoint converter input parameters

PowerPoint Converter Requirements

The PowerPoint converter requires a dedicated NT server. The dedicated NT server must have:

- a complete installation of Microsoft Office 2000
- access to a shared drive or mounted drive

Both the server that hosts the Preprocessor and the Microsoft PowerPoint converter require access to a shared drive or a mounted NFS file system.

If there are firewalls between the two processors, the specified port must be open between these machines. The Preprocessor locates the dedicated PowerPoint converter NT server by IP address and port number. The default port number is `8123`.

PowerPoint Converter Components

The Microsoft Microsoft PowerPoint converter consists of the following components:

Component	*Component Name*	*Description*
PowerPoint Converter Client	ConverterClient	Java client on the PowerPoint converter host
PowerPoint Converter Service	ConverterService	translates PowerPoint documents

Important: The PowerPoint converter uses these modules as libraries; you cannot operate the modules independently of the Preprocessor.

PowerPoint Preprocessing Overview

When the Preprocessor detects Microsoft PowerPoint documents to process, it:

- copies the PowerPoint documents to the shared location
- starts the PowerPoint converter on the NT server
- continues processing other document types

The Microsoft PowerPoint converter on the NT server:

- gets a list of documents to process
- executes the PowerPoint translation module
- returns InQuira Input XML documents to thePreprocessor

The Preprocessor then:

- integrates the InQuira Input XML documents into the preprocessing stream
- completes the remaining preprocessing steps

Copying PowerPoint Documents to the Specified Location

The Preprocessor invokes a housekeeper program that copies the PowerPoint documents from the Content Store to the specified shared location.

Translating PowerPoint Documents to XML

The `ConverterClient` passes a list of PowerPoint documents to the PowerPoint converter. The input file can be either:

- a single document
- a master file that contains a list of documents

The PowerPoint converter uses OLE objects to:

- open each document
- analyze the document object hierarchy
- translate the document to InQuira Input XML

Note: The PowerPoint converter ignores embedded OLE components, embedded Visio diagrams, and other images.

Returning the XML Files to the Preprocessor

The PowerPoint converter contains an output module that writes the translated documents to the specified output files in InQuira Input XML.

Converter Return Codes

When the PowerPoint converter completes, it issues one of the following return codes to the Preprocessor:

1	The task completed successfully.
1000	The task completed with errors. You can determine which files were processed with errors by examining the log file.
2000	The task failed before completion, and the service control terminated the master program.

Converter Error Codes

The PowerPoint converter returns the following error codes to the Preprocessor:

Error Code	*Description*
100	The Microsoft PowerPoint converter could not access the registry key `HKEY_LOCAL_MACHINE/SOFTWARE`.
200	The Microsoft PowerPoint converter could not access the registry key `HKEY_LOCAL_MACHINE/SOFTWARE/Inquira`.
300	The Microsoft PowerPoint converter could not query the registry for the value logfile under the InQuira entries.
400	The Microsoft PowerPoint converter could not open the log file specified by the logfile entry.
500	The Microsoft PowerPoint converter could not query the registry for the value config under the InQuira entries.
600	The Microsoft PowerPoint converter could not query the registry for the value port under the InQuira entries.

700	The Microsoft PowerPoint converter could not initialize the windows socket library.
800	The Microsoft PowerPoint converter could not open a socket at the port specified in the port entry.

Installing the PowerPoint Converter

You can install the PowerPoint converter as a service on the remote NT server so that it will start when the system boots. You can also enter it as a startup program under a specific login ID. When it is installed and started, the PowerPoint converter waits for client connections on a known port.

Specifying the PowerPoint Converter in the Preprocessor Configuration

To specify the PowerPoint converter for the Preprocessor, set the following parameters in the Preprocessor configuration, and in the converter configuration, `c:\_listener.xml`.

Converter name

Specify the converter name. The default value is `msoffice`.

Host name

Specify the host name of the NT server.

Port name

Specify the port on which the PowerPoint converter installed on the NT server will listen. The default is `8123`.

Configuration file

Specify the PowerPoint converter configuration file location on the NT server. The default is `c:\_listener.xml`.

Creating Custom Converters

You can create custom converters and register them with the Preprocessor in order to:

- preprocess document types for which no default converters exist
- override default converters

You can create custom convertersas Java or Perl programs.

You can use the example Perl wrappers provided in the `src/prep/examples` directory.

Creating Custom Converters in Java

You can create a custom converter in Java by implementing the Converter interface:

```
public interface Converter {
public File convert( String type, Document d ) throws
ConverterException;
}
```

where:

type

is a document type, indicated by file extension.

Document d

is a document and associated attributes, as represented in the Content Store.

The Converter interface contains a single method, convert. The convert method requires a document type and a document, and returns an IQXML document.

Implementing Converter Cleanup

You can call the Preprocessor destroy method in your custom converter if you need to perform any cleanup operations. The Preprocessor calls the destroy method to inform converters that preprocessing is complete.

```
public void destroy();
```

Registering Java Converters

You register custom Java converters in the Preprocessor configuration. To register a converter:

- specify the converter in the Preprocessor configuration, as described in ***Specifying a Custom Converter in the Preprocessor Configuration*** on page 75
- add the converter to the Preprocessor converter list, as described in ***Defining a Custom Converter within the Preprocessor Configuration*** on page 76
- define a new document type in the Preprocessor configuration, as described in ***Adding a Document Type to the Preprocessor*** on page 77
- specify the document type for the converter, as described in ***Specifying the Document Type for the Converter*** on page 77

Specifying a Custom Converter in the Preprocessor Configuration

To specify the custom converter in the Preprocessor configuration, enter appropriate values for the following parameters in the application configuration:

Parameter	*Description*	*Example*
name	Specifies the name of the converter.	`<name>converter_name</name>`
description	Specifies an optional short description.	`<description>The Custom Converter<description/>`
class	Specifies the fully qualified class name for the converter.	`<class>com.inquira.prep.custom.Converter</class>`
preserve	Specifies whether the converted files should be saved on disk after preprocessing completes. The default is false.	`<preserve>false</preserve>`
tempDirectory	Specifies the temporary directory to store the converted documents.	`<tempDirectory>prep-temp</tempDirectory>`
maxErrors	Specifies the maximum number of conversion errors that will be allowed to occur before the converter fails.	`<maxErrors>3</maxErrors>`
config	Specifies additional converter configuration values, as described in ***Specifying Additional Configuration Values*** on page 76.	`<config name=”server”>localhost</config>`
docTypes	Specifies the document types that the converter is capable of converting to IQXML. See ***Adding a Document Type to the Preprocessor*** on page 77.	`<docTypes index="0" ref="dt:CSTM">`

Specifying Additional Configuration Values

You can use the config element to specify additional configuration parameters that are not present in the converter configuration template. To use the config element, implement the config method:

```
public void config( Map config ) throws IOException;
```

And specify the config element in the form:

```
<config name='config_name'>value</config>
```

where `config_name` is the parameter name, for example, 'password', and `value` is the value of the parameter.

The configuration values are passed to the converter in the form of a `java.util.Map`, in which the keys are the names that map to their specified values.

Defining a Custom Converter within the Preprocessor Configuration

To add a custom converter to the Preprocessor converter list, edit the converter list in the preprocessor configuration file to include the custom converter. Specify the custom converter in the form:

```
<converters index="integer" ref="converter_id"/>
```

where:

index=integer

specifies the iteration of the converter within the list, such that the integer specifies the converter's position in the list.

ref="converter_id"

specifies the converter id.

Example:

```
<<preprocessors index="0" id="p:preprocessor">
<name><preprocessor</name>
<description>The default preprocessor</description>
<class>com.inquira.prep.preprocessorImpl</class>
<dtd>src/prep/input.dtd</dtd>
<converters index="0" ref="c:perl"/>
<converters index="1" ref="c:excel"/>
<converters index="2" ref="c:pdf"/>
<converters index="3" ref="c:ppt"/>
<converters index="4" ref="c:help"/>
<maxErrors>3</maxErrors>
</preprocessors>
```

Adding a Document Type to the Preprocessor

If your custom converter is designed to process a document type for which there is no InQuira 6 converter, you need to define the document type within the Preprocessor. You define the document type by editing the Preprocessor configuration file.

The configuration file specifies document types in the format:

```
<documentType index="integer"
id="converter_id">document_type</documentType>
```

where:

index=integer

specifies the iteration of the converter within the list, such that the integer specifies the converter's position in the list. You must specify document types in sequence. Refer to the Configuration and Administration Interface, or the configuration file in `com.inquira.config.bootstrap`, and any custom configuration files to determine the last index in the sequence. Specify the new document type index as the next value in the sequence. This task will be easier if done through the Administration Tool.

id="converter_id"

specifies the converter id.

document_type

specifies the document type. You map physical file extensions to this document type

Example:

```
<documentType index="0" id="dt:HTML">HTML</documentType>
<documentType index="1" id="dt:PDF">PDF</documentType>
<documentType index="2" id="dt:MS-WORD">MS-WORD</documentType>
<documentType index="3" id="dt:CSTM">CSTM</documentType>
```

Specifying the Document Type for the Converter

You specify the document type for a custom converter by editing the Preprocessor configuration file. The configuration file maps defined document types to one or more physical file extensions in the form:

```
<documentType ref="converter_id" />
<extensions index="0">new_ext</extensions>
```

where:

ref="converter_id"

specifies the converter id.

extensions index

specifies the iteration of the index within the list, such that the integer specifies the index's position in the list. You must specify document type extensions in sequence.

new_ext

indicates the physical file extension to associate with the converter. You can specify multiple physical extensions within a single document type, as shown in the following example.

Example:

```
<documentType ref="dt:HTML" />
<extensions index="0">html</extensions>
<extensions index="1">htm</extensions>
<extensions index="2">HTM</extensions>
<extensions index="3">shtml</extensions>
```

Creating Custom Converters in Perl

You can create custom Perl converters that conform to the following requirements:

- the custom converter cannot modify or update command line switches or arguments. The Preprocessor requires (or includes) the conversion module as a library or module.
- the custom converter must have main subroutines that accept arguments for input file name and output file name as shown in the following example:

```
&main($file, $outfile);
```

- the last line of the script (or wrapper) must contain the following line:

```
&::registerMyScript(\&script_name);
```

where:

`script_name` is the name of your main subroutine.

- The Perl package name must be unique, and cannot conflict with default package names. A suggested convention for naming custom converters is:

```
Package <project_name>_<project_version>_<conversion_type>
```

Registering Perl Converters

To use a custom converter for preprocessing, you need to register the converter in the Preprocessor's configuration file. If the custom converter has its own configuration file, you also need to register the configuration file.

Registering Perl converters consists of the following steps:

- associating the document type with the custom converter
- associating the document type with the custom converter configuration file, if required

Specifying Converter and Configuration File Location

You can specify the converter and configuration file names as absolute paths or as file names. If you specify only a filename, the Preprocessor will scan the following locations for the modules and files:

- the `PERL5LIB` path (the Perl library path)
- `/custom`
- `/prep`

If the Preprocessor cannot locate a specified converter or configuration file, it will exit with an error message.

Note: To ensure that the Preprocessor will use the specific converter or configuration file that you intend, specify the full path names.

Associating a Custom Module with a Document Type

When you register a converter, you associate it with a specific document type. You make the association between the converter and the document type by editing the Preprocessor configuration file, prep.config.

To associate the document type with the converter, edit the Preprocessor configuration file to include a line in the following format:

```
<DOCUMENT_TYPE> <tab> <translation_module_name>
```

Important: The entries must be tab delimited. The document type must be all upper case to avoid case sensitivity-related processing errors.

Associating a Document Type with the Custom Module Configuration File

To associate the document type with the converter configuration file, edit the Preprocessor configuration file to include a line in the following format:

```
<DOCUMENT_TYPE>.CONFIG <tab> <translation_module_config_file_name>
```

Important: The entries must be tab delimited. The document type must be all upper case to avoid case sensitivity-related processing errors.

Registration Example

The following examples show various formats that you can use to register a converter.

The example converter is called `translate.pl`, the associated document type is HTML, and the converter's configuration file is called `translate.config`.

Note: You can abbreviate the document type up to the first word.

```
HTML DOCUMENT TEXT <tab> translate.pl
HTML DOCUMENT TEXT.CONFIG <tab> translate.config
```

or

```
HTML DOCUMENT <tab> translate.pl
HTML DOCUMENT.CONFIG <tab> translate.config
```

or

```
HTML <tab> translate.pl
HTML.CONFIG <tab> translate.config
```

or

```
HTML <tab> /usr/local/bin/translate.pl
HTML.CONFIG <tab> /usr/local/bin/translate.config
```

Example Perl Wrappers for Custom Modules

You can use the example Perl wrappers located in `prep/examples` to call custom converters in combination with, or independently of, the default converters. The following example wrappers are included:

Wrapper Name	*Description*
custom_html2xml.pl	This script is a wrapper for `html2xml.pl` that allows you to use a custom converters in conjunction with a registered script. The custom converter must produce output that conforms to InQuira Input XML.
custom_pdf2xml.pl	This script is a wrapper for `pdf2xml.pl` that allows you to use a custom converter in conjunction with a registered script. The custom converter must produce output that conforms to InQuira Input XML.

custom_news2xml.pl	This script is a wrapper for `news2xml.pl` that allows you to use a custom converter in conjunction with a registered script. The custom converter must produce output that conforms to InQuira Input XML.
custom_text2xml.pl	This script is a wrapper for `text2xml.pl` that allows you to use a custom module in conjunction with a registered script. The custom converter must produce output that conforms to InQuira Input XML.
afxml2afml.pl	This script simply copies files, and can be used with the Preprocessor as a registered script to introduce files that conform to InQuira Input XML.
wrapper_example.java	This script is wrapper for a java class (`exampleForWrapper`) that illustrates how to use a custom module as a registered script. The module that you wrap must produce output that conforms to InQuira Input XML.

Example Wrapper Variables

The wrapper examples pass the following variables:

Variable	*Description*
$file	Indicates the full path to the raw input files. Do not alter this value.
$uri	Indicates the document location for retrieval. This value is used in the document tag.
$ret	Indicates the return code from the processing step.
$outfile	Indicates the full path to the output file created by the processing step.

You should only interact with the file named in `$outfile`. The Preprocessor requires this file to continue the processing stream. You can copy and edit this file, however, the `$outfile` must contain your results in order for preprocessing to continue successfully.

Important: At this processing stage, the `$outfile` must comply with the InQuira Input XML format. Files that do not conform will not be validated, and will be skipped.

If you are testing your process and returning a code (other than the return code passed in `$ret`) to indicate success or failure, you must follow the same conventions used for all registered scripts. See ***Subroutine and Module Return Codes*** on page 83 for a description of the Preprocessor return codes.

Preprocessing Modules and Routines

This section lists the modules and routines that are contained within the Preprocessor.

Perl Modules

IO::File	Allows for easy handle passing, which is required by the XML::Writer.
Getopt::Std	Get the command line options
cwd	For determining output and text directories. The current directory is the default for these if all else fails.

InQuira Modules

AF_Common	Contains the log functions: msg::die, msg::warn, msg::note, msg::blip, msg::error, and msg::info.
recursxp.pl	Contains recurs::getfiles() and recurs::outfilename, which is common code for find input and output files.
preputil.pl	Contains findConfigFile, setDate, and writeDocHeader, which are general utility scripts.

Important Global Variables

$config	prep.pl sets this variable to a translation modules configuration file.
%main	A hash of the registered modules' main subroutine's code ref. &registerMyScript updates this hash.
$type	prep.pl sets this to the document type when registering scripts. It is then used by &registerMyScript to update %main.
$AFROOT	$ENV{AFROOT}
$PREP	$AFROOT/prep
$APROOT	$ENV{APROOT}
$INDEX	$ENV{INDEX}
%modules	A hash of default conversion modules.

%config A hash of translation module configuration files.

Subroutines

ReadConfigFile	Reads prep.pl's configuration file and processes the data.
FindRegisteredScript	Finds the registered conversion script of the document currently being processed. Returns code ref.
ChkScript	Used by findRegisteredScript.
SetURI	Creates the URL for a file based on the file name, input directory and other info.
InitializeVal	Reads the DTD information into memory.
FindDirectory	Determines directory locations for script parameters not provided by command line switches such as output directory or text directory.
loadRegisteredScripts	Used during initialization to find and require conversion scripts.

Subroutine and Module Return Codes

Return codes are the error codes that a program returns if other programs that it is running have errors. ThePreprocessorrecognizes the following return codes:

Code	*Meaning*	*Action*
Undef	Problem	Issues an error message about conversion script and continues to next file.
0	Problem	Issues an error message about conversion script and continues to next file.
Throw an exception	Problem	Issues an error message about conversion script and continues to next file.
1	Successful	Completes translation.
2	Empty file	Issues message to log file and continues to next file.

Preprocessor Global Variables

ThePreprocessorsets the following global variables for the translation modules:

$config

This variable specifies thePreprocessorconfiguration file name. This variable is set by configuration information in the preprocessor's configuration file. The translation modules access this information using the line:

```
my $config_file = $::config;
```

For more information on specifying the configuration file, see ***Tailoring Preprocessing*** see "Customizing the Preprocessor" on page 60.

The InQuira Input XML DTD

The following example contains the entire Extensible Markup Language (XML) document type definition (DTD) that InQuira 6 uses to ensure standardized content processing. The DTD is made up of element declarations and attributes that are associated with specific elements. For more information on XML and XML DTDs, see the World Wide Web Consortium home page at: `http://www.w3.org`.

The first section of the DTD declares the elements that are allowed in conforming documents. The second section specifies the attributes that are associated with various elements. The elements and attributes are described in more detail in the following sections.

```
<!DOCTYPE document [
   <!ELEMENT document ((super)?, (title|section|p|a|data)*)>
   <!ELEMENT section ((title|subsection|p|a|data)*)>
   <!ELEMENT subsection ((title|subsection|p|a|data)*)>
   <!ELEMENT p (#PCDATA|s|meta|a|data)*>
   <!ELEMENT data ANY>
   <!ELEMENT s ANY>
   <!ELEMENT meta ANY>
   <!ELEMENT a EMPTY>
   <!ELEMENT title ANY>
   <!ELEMENT super ANY>

   <!ATTLIST document
             uri      CDATA    #REQUIRED
             date     CDATA    #IMPLIED
             type     CDATA    #IMPLIED>
   <!ATTLIST a
             name     CDATA    #REQUIRED>
]>
```

Element Declarations

Element declarations define the structural units, or elements, that can be included in a conforming document. Elements usually correspond to semantic parts of a document. They are specified in the following format:

```
<!ELEMENT element_name content_type>
```

where:

element_name is the name of the declared element

content_type defines the content allowed within the declared element

Note: A complete description of XML syntax is beyond the scope of this document. The content types for the elements in the InQuira DTD are described in the following section.

For example:

```
<!ELEMENT document ((super)?, (title|section|p|a|data)*)>
```

specifies that the element document may contain up to one <super> tag, and any number of <title>, <section>, <p>, <a> and <data> elements.

The InQuira input DTD declares the following elements, and specifies conditions for their use:

Element Name	*Corresponds to:*	*and Contains:*
<document>	document	A document must always begin with the <document> tag. A <document> tag may contain up to one <super> tag, and any number of <title>, <section>, <p>, <a> and <data> elements.
<section>	section	A <section> tag may contain any number of <title>, <subsection>, <p>, <a> and <data> elements.
<subsection>	subsection	A <subsection> tag may contain any number of <title>, <subsection>, <p>, <a> and <data> elements.
<p>	paragraph	A <p> element may contain parsed-character-data and the tags <s>, <meta>, <a> and <data> in any number.
<data>	data	A <data> element can contain anything. Data enclosed within <data> tags is excluded from content processing.
<s>	sentence	A <s> element can contain anything.
<meta>	meta	A <meta> element can contain anything.

<a>	anchor	An <a> element must be empty. Anchor elements are used for hypertext navigation.
<title>	title	A <title> element can contain anything.
<super>	super	A <super> element can contain anything.

Attribute Descriptions

The input DTD also defines attributes of elements. Attributes are a means of associating a value of a specific type with an element. Attributes are specified in the following format:

```
<!ATTLIST element_name attribute_name type value_type
```

where:

element_name is the name of the declared element that the attribute is associated with

attribute_name is the attribute name

type is the kind of attribute being defined. There are a limited number of attribute types that can be defined in XML. **Note:** A complete description of XML syntax is beyond the scope of this document.

value_type is the value type for this attribute, which can be specified as a default value, an implied value, or a required value

For example:

```
<!ATTLIST document uri CDATA #REQUIRED >
```

specifies that the element document contains a uniform resource indicator (URI) in the form of character data (CDATA), and that it is a required value.

The attributes specified in the InQuira DTD are described in the following table:

Element	*Attribute*	*Description*
document	uri	Document elements must include a uniform resource indicator (URI). `uri` is a required attribute. For example, `<document uri="http://www.MyWebPage.com" ...`
	date	Document elements can include a date. date is an implied attribute. For example, `<document uri="http://www.MyWebPage.com" date="document_date" ...`

	type	Document elements can include the document type. type is an implied attribute. For example, `<document uri= "http://www.MyWebPage.com" date="document_date" type="html">`
<a>	name	<a> elements must include a name. For example, `<document uri="http://www.MyWebPage.com" ...`

Chapter 7

Configuring Access to Business Data

InQuira 6 accesses business data through the Preference Service, which accesses and stores business data items that it obtains from configured data sources. The data sources that the Preference Service accesses are called *Providers*, and the data items that it uses are called *Preferences*.

You can configure InQuira 6 to access business data from the following types of Providers:

- the current InQuira 6 session
- external sources, such as CRM packages

You access business data for use by the Business Rules module, which is a separate component that enables you to:

- configure the Preference Service to access Preference data from Providers
- set Business Condition within Rules based on user, session, and other context information

Business Condition within Rules enable you to tailor application response based on various aspects of:

- user identity and activity
- session activity

The Rules Engine evaluates Business Condition within Rules by comparing the values obtained by the Preference Service to the values specified within the Business Condition.

Important: You must install and configure the Business Rules module, as described in *InQuira 6 Installation Guide*, to use the Business Rules functions.

The Preference Service

The Preference Service manages the business data Preferences obtained by the configured data Providers.

InQuira 6 includes a Session Provider that is installed and configured with the application. The Session provider is designed to access only preference data for a given session; it does not store persistent preference data for use in multiple sessions. See ***The Session Provider*** on page 91 for more information.

Access to Preference Data

The Preference Service uses a key to obtain Preference values from configured Providers. Keys take the form of multiple strings concatenated by the period (`.`) character, for example:

```
string.string.string
```

The concatenated parts enable namespace traversal for data retrieval, as described in ***Hierarchical Namespace Traversal*** on page 90.

Hierarchical Namespace Traversal

The multi-part key format that the Preference Service uses to access preference data supports hierarchical name space traversal for data retrieval.

The Preference Service locates the value of keys by attempting to match a complete key name, for example, `color.car.ext`. If it fails to locate the required key, it can create and use progressively smaller subkeys, for example `color.car`, then `color`.

You can configure a preference to require exact key matching rather than hierarchical name space traversal using the `exactMatch` parameter, as described in ***Configuring Preferences*** on page 93.

Preference Types

You can configure InQuira 6 to access the following types of business data:

User context data

User context data includes user profile information, such as user name and IP address.

Session context data

Session context data includes session-dependent information, such as refering URL and pages visited.

Question context data

Question context enables the Rules Engine to dynamically set the value of a configured preference based on information in the user's question.

You can access data using InQuira 6's configured Session Provider, or using a custom Provider that you configure using the process described in ***Creating Custom Providers*** on page 92.

Session Provider Preferences on page 91 describes the user and session context preferences that are configured for the InQuira 6 Session Provider.

Business Data Providers

Providers are the means of accessing various sources of business data, which can include the web- or application server hosting the InQuira 6 session, as well as external sources of business data, such as CRM applications.

The InQuira 6 Session provider is installed and configured with the product. ***Session Provider Preferences*** on page 91 describes the individual preferences that are configured for the Session provider.

You can define and configure additional Providers using the process described in ***Creating Custom Providers*** on page 92.

The Session Provider

InQuira 6 includes a Session provider that is installed and configured with the application. The Session provider is designed to access only preference data for a given session; it does not store persistent preference data for use in multiple sessions.

Session Provider Preferences on page 91 describes the individual preferences that are configured for these providers.

Session Provider Preferences

The Session provider is installed and configured with the InQuira 6 product. The Session provider defines the following Preferences that you can include in business conditions:

Preference	Description
Request Base URL **Request Referer Page** **User Host** **User IPAddress** **User Name**	The Preference Service obtains user data from a Principal object (`javax.security.Principal`) in the request. The principal object can be created by: • a user agent, as in JSP and ASP pages • the HTTP request headers, as are present for standard login message boxes in web browsers • a login request • a custom preference handler, as described in ***Creating Custom Preferences*** on page 93
Time (Now) **Today's Date**	The Preference Service obtains time and date data from the current system time. This data is not stored in the Preference Service.

Creating Custom Providers

You can define and configure custom Providers for the application by:

- creating a custom provider that implements the Provider application programming interface (API), as described in ***The Provider API*** on page 102
- defining the custom provider to the application, as described in ***Defining a Provider in the Application*** on page 92
- configuring Preferences to use the provider, as described in ***Configuring Preferences*** on page 93

Defining a Provider in the Application

You define a Provider using the providers map in the application configuration file. The providers map includes an entry for each provider defined for the application. Providers that are defined in the map are then available to individual preferences, as described in ***Configuring Preferences*** on page 93.

The following example shows the default Session provider configuration in the providers map:

```
<map name="providers" type="PreferenceProvider" />
<entry name="sessionProvider" type="PreferenceProvider" />
```

where:

providers	specifies the map that defines the Preference providers. You must have at least one session provider entry in this list. The Session provider is defined by default for the application.
sessionProvider	is an entry that refers to the default sessionProvider. Each defined provider must be specified as an entry in this map.

Creating Custom Preferences

You can define and configure custom business data preferences for the application by:

- creating a custom Preference that implements the Preference application programming interface (API), as described in ***The Preference API*** on page 97
- adding the Preference to the application, as described in ***Configuring Preferences*** on page 93

Configuring Preferences

You can configure Preferences for the Preference Service using the Configuration and Administration Interface. The Preferences page of the Configuration and Administration Interface accesses a preferences map within the application configuration file that specifies parameters for each defined Preference.

The following example shows the format of the preferences map:

```
<map name="preferences"
controller="com.inquira.preference.PreferenceController">
<entry name="key" type="string" />
<entry name="traversalType" type="integer" />
<entry name="exactMatch" type="boolean" />
<list  name="providers" type="PreferenceProvider" />
<entry name="purpose" type="string" />
<entry name="dataTypeClass" type="string" />
</map>
```

The Preference parameter elements correspond to the used to specify Preference parameters in the Configuration and Administration Interface, as described in ***Adding and Modifying Context Variable Preference Parameters*** on page 96.

where:

preferences is the preferences map that defines each the individual Preference.

key specifies the key required to access this Preference, for example, `color.car.ext`

TraversalType specifies how to traverse the hierarchical name space for keys that contain multiple parts when multiple providers are defined for this preference. Valid values are:

0

Exhaust the name space in the first provider before asking the next provider

1

Try the full key on all providers before processing subkeys

exactMatch is a Boolean value that specifies whether the full key is required to find the Preference. Valid values are:

- true - the full key is required
- false - the full key is not required

providers specifies the Providers that the application can use to access data for this Preference. You must specify at least one provider to configure a Preference. The Preference Service processes the list of providers in the order specified.

purpose specifies whether the Preference is accessible from the Business Rules module. Specify `BusinessCondition` to make the Preference available to the Business Rules module.

dataTypeClass	specifies the data type for this Preference, as defined in the Business Condition module. Valid values are: • `date` • `time` • `integer` • `real` • `string`
modifiable	specifies that the value of the parameter can be changed dynamically by the Business Condition module. See *InQuira 6 Language Guide* for more information on creating Business Condition that set Preference values based on question context information.

Editing Preferences Using the Configuration and Administration Interface

You can create and modify preferences using the Context Variables section of the Configuration and Administration Interface.

To edit preferences:

- select Context Variables from the Configuration and Administration Interface menu

The Context Variable page displays.

The Context Variables Page

The Context Variables page displays the currently defined context variables available to the Business Condition module. Each context variable corresponds to a configured preference that has a defined purpose of `BusinessCondition`.

Displaying the Editable Context Variable List

To display context variables as selectable items:

- select the Edit button on the Context Variables page

The Editing:Context Variables page displays.

The Editing:Context Variables Page

The Editing:Context Variables page displays the list of defined context variables as hypertext links, and displays an Add New Item link.

Selecting a Context Variable for Editing

You can select an existing context variable to modify its preference criteria, or define a new context variable to correspond to a custom preference that implements the Preference API as described in ***The Preference API*** on page 97.

To modify an existing preference:

- select the corresponding context variable from the list

To define a new preference and context variable:

- select Add New Item from the list

The Editing:Context Variables:Variables page displays.

The Editing:Context Variables:Variables Page

The Editing:Context Variables:Variables page displays fields that define the preference for the selected variable.

Adding and Modifying Context Variable Preference Parameters

You can use the fields on the Editing:Context Variables:Variables page to add or modify preference parameters for the selected context variable.

- specify values for the parameters as described in ***Configuring Preferences*** on page 93
- select OK to update the application configuration

The Preference API

You can create a preference handler by implementing the Preference API. The Preference API is defined in `com.inquira.common.Prefs`. It includes the following methods:

The getPreference Method

Method: ***getPreference***

Syntax:

```
public Preference
getPreference(java.lang.String key)
```

where:

key specifies the preference name

Description: This method retrieves the preference definition for a key from the configuration service. If the preference has not been defined, a dynamic preference object is created and returned.

Returns: the Preference object that corresponds to the requested key

The getExternalString Method

Method: getExternalString

Syntax:
```
public java.lang.String
getExternalString(java.lang.String key)
```

where:

key specifies the preference name

Description: This method returns the string representation of the preference value as defined by an external provider.

Returns: returns the string value for the specified preference

The getExternalCode Method

Method: getExternalCode

Syntax:
```
public java.io.Serializable
getExternalCode(java.lang.String key)
```

where:

key specifies the preference name

Description: This method returns the native form of the preference value as defined by the provider. This can be null if no key access using this preference value is required that is based not on the external string.

Returns: the code value for specified preference

The getPreferenceValue Method

Method: getPreferenceValue

Syntax:
```
public PreferenceValue
getPreferenceValue(java.lang.String key)
```

where:

key specifies the preference name

Description: This method returns the `PreferenceValue` object that corresponds to this key.

Returns: the preference value

The SetPreferenceValue Methods

There are several `setPreferenceValue` methods within the Preference API. The various `setPreferenceValue` methods use various combinations of `key`, `value`, `code`, `string`, and `preference` parameters.

Method: setPreferenceValue

Syntax:

```
public PreferenceValue
setPreferenceValue(java.lang.String key,
java.lang.String value)
```

where:

key specifies the preference name

value specifies the string value

Description: This method returns the `PreferenceValue` object that corresponds to this key.

Returns: the updated `PreferenceValue` object

Method: setPreferenceValue

Syntax:

```
public PreferenceValue
setPreferenceValue(java.lang.String key,
java.io.Serializable code)
```

where:

key specifies the preference name

Description: This method sets the code for the preference to the `code` argument, the string value to `code.toString( )` with the given key, and returns the updated `PreferenceValue` object. If no preference with this key exists, it creates a new preference using only the session preferences.

Returns: the updated `PreferenceValue` object

Method: setPreferenceValue

Full Name:

```
public PreferenceValue
setPreferenceValue(Preference preference,
java.lang.String value)
```

where:

key specifies the key value

value specifies the string value

Description: This method sets the value for the preference with the given key and returns the updated `PreferenceValue` object. If no preference with this key exists, it creates a new preference using only the session preferences.

Returns: the updated `PreferenceValue` object

Method: setPreferenceValue

Syntax:

```
public PreferenceValue
setPreferenceValue(Preference preference,
java.lang.String value,
java.io.Serializable code)
```

where:

preference specifies the preference

value specifies the string value

code specifies the code value

Description: This method sets the value and code for the preference with the given preference and returns the updated `PreferenceValue` object. If no preference with this key exists, it creates a new preference using only the session preferences.

Returns: the updated `PreferenceValue` object

Method: setPreferenceValue

Syntax:

```
public PreferenceValue
setPreferenceValue(java.lang.String key,
java.lang.String value,
java.io.Serializable code)
```

where:

value specifies the string value

code specifies the code value

Description: This method sets the value and code for the preference with the given key and returns the updated `PreferenceValue` object. If no preference with this key exists, it creates a new preference using only the session preferences.

Returns: the updated `PreferenceValue` object

Method: setPreferenceValue

Syntax:

```
public PreferenceValue
setPreferenceValue(PreferenceValue value)
```

where:

value specifies the preference value

Description: This method sets the value for the preference with the given key and returns the passed `PreferenceValue` object. If no preference with this key exists, it creates a new preference using only the session preferences.

Returns: the updated `PreferenceValue` object

The rundown Method

Method: rundown

Syntax:

```
public void rundown() throws TransportException
```

Description: The rundown method will be called after every completion of a request from the `Execution.rundown()` method.

Throws: `TransportException`

The getPreferences Method

Method: ***getPreferences***

Syntax: `public java.util.Collection getPreferences()`

Description: This method retrieve a list of all defined preferences. The returned list contains `Preference` objects.

Returns: A list of all `Preference` objects.

The getPreferenceKeys Method

Method: ***getPreferenceKeys***

Syntax: `public java.util.Collection getPreferenceKeys()`

Description: This method retrieve a list of all defined preference keys. The returned list contains `String` objects.

Returns: a list of all `String` objects

The Provider API

You can create a custom provider to access business data by implementing the Provider API. Providers must extend the class `com.inquira.preference.AbstractProvider`, which includes the following methods:

The Provider Default Constructor

Method: ***AbstractProvider***

Syntax: `public AbstractProvider()`

Description: Specifies the default constructor. A default constructor is required so that the class can be loaded dynamically.

The getType Method

Method: ***getType***

Description: This method should not be overridden. The static `String` variable type should be initialized in a static initializating code block. For example:

```
static {type = "my type";}
```

Returns: the provider type as a string

The get Methods

There are two `get` methods within the Provider API; a namespace hierarchy traversal method, and a non-traversal method.

Name: ***get***

Syntax:

```
public PreferenceValue get(java.lang.String key,
byte depth, byte traversalDepth)
```

where:

key specifies the preference name

depth specifies the number of elements in the key

traversalDepth specifies the number of elements that can be removed from the key to create subkeys

Description: This method retrieves the value that corresponds to the key argument within this provider. The number of key elements is specified by the `depth` parameter.

This function will traverse the namespace hierarchy for the key by attempting to locate subkeys if the full key location fails. Subkeys are created by removing the final key element; for example:

```
color.car.ext -> color.car -> color
```

You can limit namespace traversal by specifying a traversal depth argument. This argument specifies the number of key elements that can be removed to create subkeys. For example:

```
key = color.car.ext
depth = 3
traversalDepth = 1
```

specifies that the only valid keys are `color.car.ext` and `color.car`.

`get` is dependent on the `get(String)` method, which must be implemented by all subclasses. This method does not have to be overridden in custom providers unless the functionality is not required.

Returns: a `PreferenceValue` object with the appropriate values set

`null` if no match is found

Method: get

Syntax:

```
protected abstract PreferenceValue
get(java.lang.String key)
```

where:

key specifies an exact key name

Description: This abstract method is called by `get(String,byte,byte)` to find the value for a given key. This method does not create and match subkeys to traverse the key namespace; it only checks for an exact key match. This method must be defined by all provider classes and must implement the specific functionality for this provider.

Returns:

- the `PreferenceValue` object, if it locates the exact key
- `null`, if it does not locate the exact key

The set Method

Method: ***set***

Syntax:
```
protected abstract boolean
set(java.lang.String key, PreferenceValue value)
```

where:

value specifies the preference value

Description: This abstract method sets the value for the implied preference of the value object.

Returns: the preference value

The toString Method

Method: ***toString***

Syntax: `public java.lang.String toString()`

Description: This method returns information about the provider that can be useful for debugging.

Overrides: `toString` in the class `java.lang.Object`

Returns: information about the provider

The Provider rundown Method

Method: ***rundown***

Syntax: `public abstract void rundown()`

Description: This method is called when the preference services' `rundown` method is called. This method should store and retain preference values for this provider.

Chapter 8

Configuring Document Security

You can configure InQuira 6 to permit only authorized users to access answers from source documents. InQuira 6 calls the configured document security function prior to presenting answers in the User Interface, so that the application will display information in protected documents only to authorized users. You can implement document security using:

- the default implementation installed with the product, as described in ***The Default Document Security Implementation*** on page 107.
- a custom implementation, as described in ***Implementing a Custom Document Security Function*** on page 111.

Note: The document security feature applies only to answers retrieved by the unstructured information retrieval module.

The Default Document Security Implementation

The default document security implementation uses Lighweight Directory Access Protocol (LDAP) to verify user permision for document access. InQuira 6 calls the document security function prior to presenting answers in the User Interface.

The default implementation verifies user permission by:

- searching the LDAP directories for user and document role information
- comparing assigned user roles with assigned document roles

Note: You must have LDAP directories available for use by the document security function.

The document security function searches for users by username, and for documents by URL. It performs a security check for each document associated with an answer returned by the unstructured information retrieval module.

The document security function permits answers only if the user and document share at least one role, as defined in the LDAP directories. The document security function eliminates unauthorized answers from the User Interface answer page.

Configuring the Default Security Implementation

InQuira 6 is installed with a default document security implementation. You can configure the default security implementation for your application by:

- enabling the default security implementation
- specifying an optional anonymous user name
- specifying LDAP user and document directory locations
- configuring application access to the specified LDAP directories
- specifying LDAP access parameters for the application

You configure the default document security implementation from the Excerpt Service page of the Configuration and Administration Interface. To access the Excerpt Service page from the Configuration and Administration Interface menu:

- select Excerpt Service

The Excerpt Service page displays the Security area.

Enabling the Default Security Implementation

You enable the application to perform user and document security checking by selecting the Enable Security radio button in the Security area of the Configuration and Administration Interface Excerpt Service page.

Specifying an Anonymous User

You can specify a user name to apply to anonymous users for the purpose of security checking. The name that you specify must have a corresponding LDAP entry.

To specify an anonymous user name:

- select Edit on the Excerpt Service page

The Excerpt Service page displays the Security options.

- enter the user name
- select OK to save your configuration

Specifying an LDAP Document Directory

You can specify the LDAP directory that contains the document security information that you want to use for the application.

To specify a document directory:

- select Edit on the Excerpt Service page

The Excerpt Service page displays the Security options.

- enter the LDAP directory name
- select OK to save your configuration

Specifying an LDAP User Directory

You can specify the LDAP directory that contains the user security information that you want to use for the application.

To specify a user directory:

- select Edit on the Excerpt Service page

The Excerpt Service page displays the Security options.

- enter the LDAP directory name
- select OK to save your configuration

Specifying LDAP Access Parameters

You can configure additional LDAP access parameters as required by your LDAP implementation.

To specify a LDAP access paramters:

- select Edit on the Excerpt Service page
- select the Advanced option from the drop-down menu

The Excerpt Service page displays the advanced Security options.

- specify the following access parameters for the application:
  - authentication method
  - principal
  - credentials
  - security protocol
  - permission attribute

Note: The access parameters are standard elements of an LDAP implementation. Consult your LDAP administrator for assistance in determining the correct calues for these parameters.

- select OK to save your configuration

Example Security Configuration

The following shows an example of the security configuration for an application:

```
<security>
    <use-security>false</use-security>
    <security-class/>
    <ldap-security>
        <user>
             <provider-url>ldap://localhost:389</provider-url>
             <authentication-method>simple</authentication-method>
             <principal>ldapuser</principal>
             <credentials>mypassword</credentials>
             <security-protocol>ssl</security-protocol>
             <permission-attribute>role</permission-attribute>
        </user>
        <document>
             <provider-url>ldap://localhost:389</provider-url>
             <authentication-method>simple</authentication-method>
             <principal>ldapuser</principal>
             <credentials>mypassword</credentials>
             <security-protocol>ssl</security-protocol>
             <permission-attribute>role</permission-attribute>
        </document>
    </ldap-security>
</security>
```

Implementing a Custom Document Security Function

You can create a custom document security function by implementing the document security API, as described in ***The Document Security API*** on page 111.

The Document Security API

The Document Security API consist of a single class, `com.inquira.security.ISecurity`. This class contains the following methods:

- ***The hasPermission Method*** on page 111
- ***The anonymousUser Method*** on page 111

The hasPermission Method

Method: ***hasPermission***

Syntax: `public boolean hasPermission(String username, String url) throws SecurityException`

where:

username is a user name

url is a URL

Description: This boolean method evaluates permission based on username and URL objects and returns boolean value `true` or `false`.

Returns: a boolean value

The anonymousUser Method

Method: ***anonymousUsername***

Syntax: `public String anonymousUsername() throws SecurityException`

Description: This method assigns a specified value to an unknown user.

Returns: the specified string

Example:

```
public String anonymousUsername() throws SecurityException
    {  return "guest"; }
```

Document Security Implementation Example

The following example is a simple implementation of the document security API that implements the interface `{@link com.inquira.security.ISecurity}`.

This implementation assumes a simple web site consisting of:

- public pages
- private pages

Multiple users have access to all public pages and to their own private pages.

The URLs of the private pages follow the form:

`http://host/~<user>...`

where:

`<user>` is the user name.

```
package com.inquira.security;
import java.net.*;
import com.inquira.security.*;

public class SimpleSecurityImpl
    implements ISecurity
{
    public boolean hasPermission( String username, String url )
        throws com.inquira.security.SecurityException
    {
        boolean rc = false;
        // If the user name is not set use the anonymous user name for
checks.


        if( username == null || ( username = username.trim( ) ).length( )
== 0 ) {
        username = anonymousUsername( );
        }
        try {
            // obtain the path element of the URL
            URL u = new URL( url );
            String path = u.getPath( );
            if( path != null ) {
                if( path.startsWith( "/~" ) ) {
                    if( username != null &&
                        ( path.startsWith( "/~" + username + "/" ) ||
```

```
path.equals( "/~" + username ) ) ) {
                        // allow access to the users own files
                        rc = true;
                    }
                } else {
                    // allow access to the public pages
                    rc = true;
                }
            } else {
                // allow public access to the sites entry point
(http://host)
                rc = true;
            }
        } catch( MalformedURLException ex ) {
            throw new com.inquira.security.SecurityException(
"MALFORMED_URL", new Object[]{ url } );
        }
        return rc;
    }
    public String anonymousUsername( )
        throws com.inquira.security.SecurityException
    {
        return "guest";
    }
    /** This simple main program test the implemenation of the security
API.
    * <BR>Examples:<BR>
    * <PRE>
    * $ java SimpleSecurityImpl "user1" "http://host/~user1"
    * hasPermission( "user1", "http://host/~user1" ) = true
    *
    * $ java SimpleSecurityImpl "user1" "http://host/~user1/file.html"
    * hasPermission( "user1", "http://host/~user1/file.html" ) = true
    *
    * $ java SimpleSecurityImpl "user1" "http://host/~user2/file.html"
    * hasPermission( "user1", "http://host/~user2/file.html" ) = false
    *
    * $ java SimpleSecurityImpl "user1" "http://host"
    * hasPermission( "user1", "http://host" ) = true
    *
    * $ java SimpleSecurityImpl "user1" "http://host/file.html"
    * hasPermission( "user1", "http://host/file.html" ) = true
    *
    * $ java SimpleSecurityImpl "" "http://host/~user2/file.html"
    * hasPermission( "", "http://host/~user2/file.html" ) = false
    * </PRE>
    */
   public static void main( String[] args )
        throws Exception
   {
        SimpleSecurityImpl ssi = new SimpleSecurityImpl( );
        for( int i = 0; ( i + 1 ) < args.length; i += 2 ) {
            String user = args[i];
```

```
            String url  = args[i+1];
            System.out.println( "hasPermission( \"" + user + "\", \"" + url
+ "\" ) = " +
                   ssi.hasPermission( user, url ) );
            }
        }
    }
```

Chapter 9

Scheduling InQuira 6 Processes

You can schedule various InQuira 6 operations by defining and scheduling jobs, using the Scheduler. You access the Scheduler through the Configuration and Administration Interface.

The scheduler contains pre-defined tasks that you combine into task sets and schedule as jobs. You can schedule jobs as either single or recurring events.

To schedule a job, you need to:

- define or select a task set, as described in ***Defining Task Sets*** on page 115
- specify one or more collections for the job, as described in ***Specifying Target Collections*** on page 124
- specify scheduling parameters for the the job, as described in ***Specifying Task Set Recurrance*** on page 116
- specify email notification for the job

Defining Task Sets

Task sets are collections of tasks that you can define as a single logical entity. A single task set can contain multiple related tasks, such as crawling, preprocessing, and indexing application content. You can include one or more tasks in a task group.

You define a task set by specifying the following parameters:

Parameter	*Description*
Name	Specifies a name for the task set. **Note:** Each task set must have a unique name.
Description	Specifies an optional description for the task set.
Enabled	Specifies the job's activation status. See ***Specifying Activation Status*** on page 116.

Recurring Type	Specifies the type of interval for recurring tasks. See ***Specifying the Interval Type*** on page 116.
Recurring Value	Specifies the number of intervals to elapse between task events. See ***Specifying the Interval Value*** on page 117.
Task List	Specifies the tasks to perform as part of the defined task set. See ***Specifying Tasks for a Task Set*** on page 119.
Collection List	Specifies the defined content collections that are targets of this task set. See ***Specifying Target Collections*** on page 124.

Specifying Activation Status

You can specify whether a defined task set is active and eligible for execution, or disabled. The scheduler will not execute disabled task sets. You can disable a task set to suspend it from processing while saving its definition so that you can later enable it without re-defining it. Task sets are enabled by default.

To enable a task set:

- select the On radio button in the Enabled field

The scheduler will execute the task set at the specified time and interval.

To disable a task set:

- select the Off radio button in the Enabled field

The scheduler will suspend execution of the task set, but the task set definition will remain available.

Specifying Task Set Recurrance

You can specify whether the job will recur at a defined interval, or be executed once only at the defined start time. You specify the basis upon which the task set will recur by selecting an interval type, as described in ***Specifying the Interval Type*** on page 116, and an interval value, as described in ***Specifying the Interval Value*** on page 117.

Specifying the Interval Type

You can specify the following types of task set intervals:

- Monthly

- Weekly
- Daily
- Hourly
- Once

To specify a task set interval:

- select the interval type from the drop-down menu

The scheduler will execute the job at the specified interval, beginning at the specified start time. You can specify the number of intervals that will elapse between task executions using the Recurring Value field, as described in ***Specifying the Interval Value*** on page 117.

Specifying the Interval Value

The task set interval value specifies the number of intervals that will elapse between task executions. For example, if you specify an interval of Weekly, and a value of 2, the job will execute every two weeks at the specified start time.

To specify a task interval value:

- enter a value in the Recurring Value field. The value can be any integer.

The scheduler will execute the job at the specified interval, beginning at the specified start time.

Specifying Email Notification for a Task Set

You can specify email notification of task completion status to one or more recipients using the Scheduler's email notification facility. The email notification facility sends messages to specified users to inform them of task status.

You use the Scheduler's email notification facility by:

- specifying an email sender for the application, as described in ***Specifying an Email Sender*** on page 118
- defining one of more groups of email recipients, as described in ***Defining Email Recipient Groups*** on page 118

The email messages contain information from the log messages associated with the task completion status.

Specifying an Email Sender

You must define an email sender for the Scheduler's email notification facility. You must specify the address of a valid email account on the email server configured for the application. To specify an email sender for the application:

- select Scheduled Tasks from the Configuration and Administration Interface

The Distributed Scheduler page displays.

- select Edit on the Distributed Scheduler page

The Editing: Distributed Scheduler page displays.

- enter a valid email address in the Email User field, and select OK to update the application configuration

Defining Email Recipient Groups

You must create one or more email recipient groups for the Scheduler's email notification facility. Email recipient groups contain one or more valid email addresses. The Scheduler sends notification messages to the addresses within the group when the associated task set completes.

You define an email recipient group by:

- specifying a group name, as described in
- specifying group member addresses, as described in

Specifying a Group Name

You specify a name for an email recipient group using the E-Mail Group Name field of the Editing: distributedscheduler > E-Mail Group > E-Mail Group page in the Configuration and Administration Interface. To specify a group name:

- select Scheduled Tasks from the Configuration and Administration Interface

The Distributed Scheduler page displays.

- select Edit on the Distributed Scheduler page

The Editing: Distributed Scheduler page displays.

- select the Edit List link of the Notify field

The Editing: Distributed Scheduler > E-Mail Group page displays.

- select the Add New Item link of the E-Mail Group field

The Editing: Distributed Scheduler > E-Mail Group > E-Mail Group page displays.

- enter a name in the E-Mail Group Name field, and select OK to update the application configuration

Specifying Group Members

You specify members of an email recipient group using the Addresses field of the Editing: Distributed Scheduler > E-Mail Group > E-Mail Group > E-Mail Addresses. You must specify at least one member for each defined group.

To specify a group name:

- select Scheduled Tasks from the Configuration and Administration Interface

The Distributed Scheduler page displays.

- select Edit on the Distributed Scheduler page

The Editing: Distributed Scheduler page displays.

- select the Edit List link of the Notify field

The Editing: Distributed Scheduler > E-Mail Group page displays.

- select the Add New Item link of the E-Mail Group field

The Editing: Distributed Scheduler > E-Mail Group > E-Mail Group page displays.

- select the Add New Item link of the E-Mail Addresses field

The Editing: Distributed Scheduler > E-Mail Group > E-Mail Group > E-Mail Addresses page displays.

- enter an address in the Addresses field and select OK to update the application configuration. Repeat the process to add member addresses as needed.

Specifying Tasks for a Task Set

You can specify one or more tasks within a task set; the available tasks are described in:

Content Acquisition Tasks

InQuira 6 defines the following content acquisition tasks:

Task	*Description*
Content Status Override	Use this task to override a failed status for acquisition, preprocessing, and indexing, and proceed with the next content processing step. For example, you can execute a task to override content acquisition failure due to an exceeded variance threshold for a collection. See ***Specifying Variance Thresholds for Collections*** on page 14 for more information on setting document count variance thresholds for collections.
Content Reset	Use this task to reset, or delete, the documents within the specified collections. The specified collections will be marked as empty until populated by the next content acquistion process.
Content Update	Use this task to update all documents within the specified collections. The content (documents) defined for the specified collections will be crawled and updated according to the current crawl parameters.
Content Baseline	Use this task to set the acquisition baseline to the current document count. For example, you can execute a task to reset the content baseline for a collection that failed acquisition due to an exceeded variance threshold. See ***Specifying Variance Thresholds for Collections*** on page 14 for more information on setting document count variance thresholds for collections.

Content Processing Tasks

InQuira 6 defines the following content processing tasks:

Task	Description
Content Update	Use this task to update all documents within the specified collections. The content (documents) defined for the specified collections will be crawled and updated according to the current crawl parameters.
Preprocess Incremental	Use this task to preprocess only documents that have been modified since the last preprocessing task.
Preprocess Full	Use this task to preprocess all documents defined for the specified collection.
Index Incremental	Use this task to index only documents defined for the specified collections that have been modified since the last indexing task.
Index Full	Use this task to index all documents defined for the specified collection.
Index Maintenance	Use this task to rebuild index cache files after the indexing process completes as required for page rank calculations and distributed content processing. **Note:** This task is optional; if you have not configured distributed content processing, and do not want page rank calculation to influence unstructured content retrieval, you do not need to execute this task.

Content Metadata Tasks

InQuira 6 defines the following content metadata-related tasks:

Task	Description
Supertitles	Use this task to apply updated document supertitle metadata to the specified collections. The application will apply the supertitle changes to the content when you execute an Index task.

Attributes	Use this task to apply updated document supertitle metadata to the specified collections. The application will apply the supertitle changes to the content when you execute an Index task.

Content Database Tasks

InQuira 6 defines the following content database-related task:

Task	*Description*
Content Schema Re-Create	Use this task to re-create the Content Store database tables. This task will delete the existing Content Store data and create a new Content Store database containing no data.

InQuira 6 Analytics Tasks

InQuira 6 defines the following InQuira 6 Analytics-related tasks that you can combine into task sets for scheduling:

Task	*Description*
ODS Schema Re-Create	Use this task to re-create the InQuira 6 Analytics ODS tables. The ODS tables store the direct representation of the application run-time logs. You can re-create these tables at any time, for example, to minimize the disk space dedicated to InQuira 6 Analytics processing, provided that the Analytics Cube Creation task has processed any data that will be deleted when the application re-creates the tables. For example, you could schedule the following tasks in sequence to re-create the ODS tables: • Log Extraction task • ODS Schema Re-Create • Analytics Load • Analytics Cube Creation The application will delete the existing ODS data and create a new ODS containing no data.

STAR Schema Re-Create	Use this task to re-create the InQuira 6 Analytics reporting database tables. The reporting database is a star-schema, which is based on the original ODS tables, but is organized into aggregated, historical data structures that are optimized for analytical applications. **Important:** The application uses the reporting data within the current available reports. If you delete these tables, the existing reports will stop functioning. The application will delete the existing reporting data and create a new reporting database containing no data.
Analytics Load	Use this task to download the log files from the production servers to the InQuira 6 Analytics server. The application stores a copy of each log file in: • the InQuira 6 Analytics server's `data/load` directory, or the directory specified in the InQuira 6 Analytics configuration. The files in this directory are available to the Analytics Load task. The Analyltics Load task deletes the copy in the data/load directory upon successful processing. • the InQuira 6 Analytics server's `data/archive` directory, or the directory specified in the InQuira 6 Analytics configuration. This directory stores a compressed version of the file for use by optional tape backup processes.
Analytics Cube Creation	Use this task to create the InQuira 6 Analytics report data cubes using the current reporting database. The report data cubes are the set of files derived from the reporting (star-schema) database that InQuira 6 Analytics uses to generate the report displays. See *InQuira 6 Analytics Guide* for more information on report data cubes.
Analytics Complete Processing	Use this task to perform the Analytics Load and Anlytics Cube Creation tasks as a single task.

Analytics Error Reloading	Use this task to reload log files that failed to load due to an error condition. The functions of this task are incorporated into the Analytics Load task. The Analytics Load task incorporates the automatically checks for errors occurring in a previous load, and attempts to reload any portions of the log file that failed. The Analytics Load process will terminate before loasing new log files if it cannot resolve the previous error conditions. This task provides a separate, explicit means of performing the same function.

Production Control Tasks

InQuira 6 defines the following tasks that you can combine into task sets for scheduling:

Task	*Description*
Staging to Production Synch	Use this task to copy all application data from this data source to all of the data sink instances configured for the application. See *InQuira 6 Application Guide* for more information on configuring instances and instance groups.
Production Log Extraction	Use this task to copy log data from a defined data source instance. See *InQuira 6 Application Guide* for more information on configuring instances and instance groups.

Specifying Target Collections

You can define a task set to apply to multiple collections of various types. To specify the target collection(s) for a task set:

- select the desired collection type

The Editing: Scheduler > Task Set > Collections page displays a list of defined collections of the selected type.

- select the desired collections to add them to the task set
- select OK to save the task set

Specifying Start Parameters

You specify the start time for a task set using the Start Time fields on the Scheduler page. Specify the following values to define a start time:

Parameter	*Definition*
Month	Select from January to December.
Day	Select a value from 1-31.
Year	Select a value from 2003-2005.
Hour	Select a value from 00-23.