



Technical Requirements Specification

DELIVER Requirements

Steve Beech

Table of Contents:

DELIVER TECHNICAL REQUIREMENTS SPECIFICATION	2
1.0 ANGEL SMART LINK FINDER CLIENT SPECIFICATION	2
1.1 <i>Architecture</i>	2
1.2 <i>Messaging</i>	2
1.3 <i>Functionality</i>	4
1.4 <i>Process Walk-Through</i>	4
1.6 <i>Functional Decomposition</i>	5
1.7 <i>Timetable for Development</i>	6
2.0 ANGEL URL HARVESTING TOOLS SPECIFICATION.....	7
2.1 <i>Overview</i>	7
2.2 <i>Architecture</i>	7
2.3 <i>Tool Types</i>	7
2.4 <i>Functional Decomposition</i>	8
2.5 <i>Timetable for Development</i>	8

Deliver Technical Requirements Specification

The following specification has been written as part of the TRS 0.30/0.40 requirements specifications and ICT/DCT 0.10 development plans, and defines front-end tools and user interfaces in order to meet Deliver/Devil requirements as identified in the User Needs Analysis document UNA_fullrecs.xls (version1.1)

The tools defined in this specification address some of the requirement 'gaps' in DELIVER and DEVIL in the following ways:

The SLF allows for the searching of databases utilising the Angel Resource Manager and therefore relies on persistent URLs, which can be maintained centrally.

The SLF will allow for the embedding of library resources into VLEs, through searching library databases using standard bibliographical search terms.

The UHT will allow the embedding of external resources (www) into VLEs, using the URL and metadata available directly from the resource.

1.0 Angel Smart Link Finder Client Specification

This specification for the Angel Smart Link Finder has been written as part of the TRS 0.30/0.40 requirements specifications and ICT/DCT 0.10 development plans, and defines front-end tools and user interfaces in order to meet Deliver/Devil requirements as identified in the User Needs Analysis document UNA_fullrecs.xls (version1.1)

Overview

The Smart Link Finder (A-SLF) will allow a user to search for items within resources to which he has access rights, using the Angel Resource Manager (A-RM) to control access to those resources.

The A-SLF will allow for the creation of references to these items, which may be placed within a reading list for use within an institution.

1.1 Architecture

The early prototype of the client will be based on trusted Java Applet technology, which will allow the embedding of the client application into web pages. This should provide a consistent interface to the Angel RM across multiple systems, such as VLEs and MLEs.

In order to allow the applet to communicate with the A-RM and/or write to a local file it will have to be signed with a certificate in order to create a trust relationship. In the first instance a self-generated Angel certificate will be used, but this might be effectively replaced by one from a known issuer for production versions.

1.2 Messaging

The message specification needs to define messages at an item level, rather than the resource level messaging used for most of the A-RM functionality. In order to provide the messages in the most appropriate format the messages will extend upon the searchRequest and searchResponse messages as defined in the [ANGEL Messaging Specification Version 0.13 paragraph 4.11](#), using the <querystring> element to describe the BIB-1 search terms (and values) as identified below:



An example search request:

```
<searchRequest>
  <requesterIP>123.23.53.35</requesterIP>
  <searchType>bibliographic</searchType>
  <query>
    <queryString> //result[//bib1[(@field="4") and (xf::lower-case(/text()) ="java and xml")] and
    (/bib1[(@field="1003") and (xf::lower-case(/text()) ="brett mclaughlin")] </queryString>
    <maxResults>50</maxResults>
    <firstResultNo>251</firstResultNo>
  </query>
  <resourceId>345783</resourceId>
</searchRequest>
```

Example search response

```
<searchResponse>
  <diagnostic>
    <diagnosticCode>1000</diagnosticCode>
    <diagnosticString>OK</diagnosticString>
  </diagnostic>
  <result>
    <bib1 field="4">Java and XML</bib1>
    <bib1 field="1003">Brett McLaughlin</bib1>
  </result>
  <result>
    <bib1 field="4">Java and XML</bib1>
    <bib1 field="1003">Somebody Else</bib1>
  </result>
</searchResponse>
```

Search Terms:

BIB-1 Attributes

Name	Value	Semantics	Name	Value	Semantics
Personal name	1		No. govt pub.	50	
Corporate name	2		No. music publisher	51	
Conference name	3		Number db	52	
Title	4		Number local call	53	
Title series	5		Code--language	54	
Title uniform	6		Code--geographic area	55	
ISBN	7		Code--institution	56	
ISSN	8		Name and title *	57	
LC card number	9		Name geographic	58	
BNB card no.	10		Place publication	59	
BGF number	11		CODEN	60	
Local number	12		Microform generation	61	
Dewey classification	13		Abstract	62	
UDC classification	14		Note	63	
Bliss classification	15		Author-title	1000	
LC call number	16		Record type	1001	
NLM call number	17		Name	1002	



NAL call number	18	Author	1003	
MOS call number	19	Author-name personal	1004	
Local classification	20	Author-name corporate	1005	
Subject heading	21	Author-name conference	1006	
Subject Rameau	22	Identifier--standard	1007	
BDI index subject	23	Subject--LC children's	1008	
INSPEC subject	24	Subject name -- personal	1009	
MESH subject	25	Body of text	1010	
PA subject	26	Date/time added to db	1011	
LC subject heading	27	Date/time last modified	1012	
RVM subject heading	28	Authority/format id	1013	
Local subject index	29	Concept-text	1014	
Date	30	Concept-reference	1015	
Date of publication	31	Any	1016	
Date of acquisition	32	Server-choice	1017	
Title key	33	Publisher	1018	
Title collective	34	Record-source	1019	
Title parallel	35	Editor	1020	
Title cover	36	Bib-level	1021	
Title added title page	37	Geographic-class	1022	
Title caption	38	Indexed-by	1023	
Title running	39	Map-scale	1024	
Title spine	40	Music-key	1025	
Title other variant	41	Related-periodical	1026	
Title former	42	Report-number	1027	
Title abbreviated	43	Stock-number	1028	
Title expanded	44	Thematic-number	1030	
Subject precis	45	Material-type	1031	
				Original semantics (changed 8/97, see extensions below).A persistent identifier, or Doc-ID, assigned by a server, that uniquely identifies a document on that server.
Subject rswk	46	Doc-id	1032	
Subject subdivision	47	Host-item	1033	
No. nat'l biblio.	48	Content-type	1034	
No. legal deposit	49	Anywhere	1035	
		Author-Title-Subject	1036	

1.3 Functionality

The applet will collect search terms from the user, create an XRR searchRequest message, connect to the Angel RM using a raw socket connection, send the message, retrieve a searchResponse message, analyse the response to extract the relevant BIB1 data and produce URL references to the items, citing them in Harvard format. These results will initially be displayed within the applet itself, so that they may be copied and pasted as required. Later versions of the client applet will write results directly to a page within the appropriate system (i.e. VLE/MLE)

1.4 Process Walk-Through

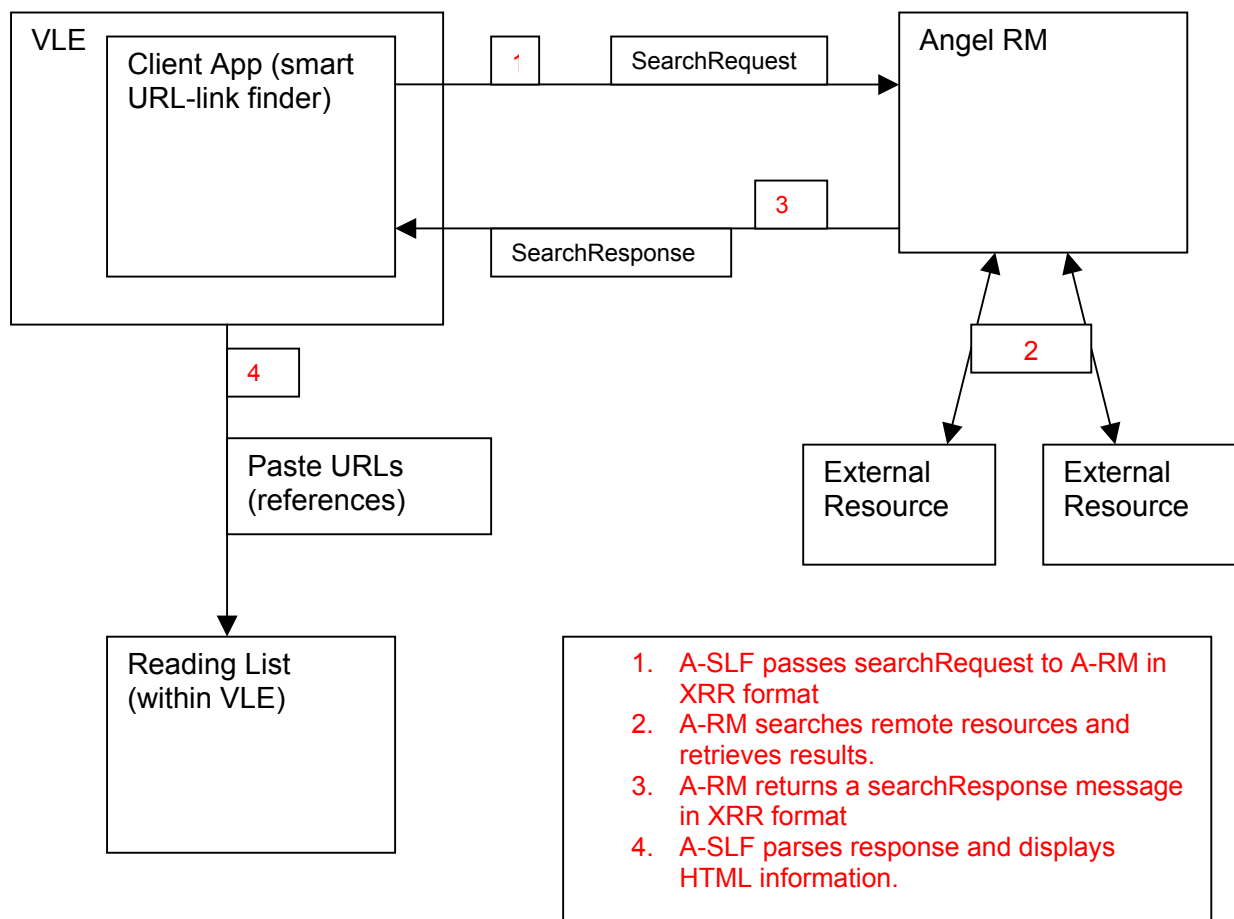
A user accesses the SLF (Smart Link Finder) from within the VLE (initially this will be demonstrated externally (from the VLE) as a standalone applet)
 Search boxes allow the user to search on a number of bibliographic fields (e.g. Title, Author etc) and pressing 'Search' initiates the search process.

The SLF sends a searchRequest message (as shown below) to the Angel RM, which searches the remote resource (specified within the searchRequest message by resourceID) and returns a searchResponse message, which contains a number of located resource items.

The SLF applet transforms these items into HTML for display (within the applet) and allows the user to copy and paste them into the reading list.

The following diagram demonstrates this process:

An outline diagram showing the proposed functionality of the SLF:



1.6 Functional Decomposition

The Smart Linker Finder development will be broken down into several discreet elements as listed below:

Search Form – initially a search form will be developed which allows for the input of various BIB-1 search terms.



Search Action – a servlet will be developed which accepts the search terms from the above form and communicates with the RM to retrieve the results (in XRR format)

Client Applet – an applet will then be developed to incorporate the above two pieces of functionality.

A List Servlet – a second servlet will be developed to accept the results from the applet and create a list of the resources identified within the search response. This will form the basis of the resource list and will later be incorporated into the required VLE programs.

1.7 Timetable for Development

The initial development will focus on functionality external to any specific institutional systems such as VLEs, MLEs etc.

Early prototypes of the deliverables will be created over the next few weeks, with an alpha system available by the end of March 2003.

Further development will then focus on adapting the functionality to integrate with institutional systems and a beta production version will be produced by the end of June 2003.

A more detailed timetable of development of each module is given below:

Code and Test Search Form – 2 days

Code and Test Search Action Servlet – 5 days

Code and Test Client Applet – 4 days

Code and Test List Servlet – 3 days

2.0 Angel URL Harvesting Tools Specification

This specification for the Angel URL Harvesting Tools (A-UHT) has been written as part of the TRS 0.30/0.40 requirements specifications and ICT/DCT 0.10 development plans, and defines front-end tools and user interfaces in order to meet Deliver/Devil requirements as identified in the User Needs Analysis document UNA_fullrecs.xls (version1.1)

2.1 Overview

To allow for effective harvesting of URLs to be included in a reading list, the development of URL harvesting tools is required.

These tools will allow a user to browse to a chosen website (that they would like to include within their reading list, and by simply clicking a link within their browser toolbar, add the title and URL of the page to the reading list.

Initial development of these tools will produce standalone functionality based upon the use of Javascript and a web server hosting JavaServer Pages. This should allow for an easy integration path to inclusion within VLEs at a later date.

2.2 Architecture

The early prototype tools will be based upon http links with embedded JavaScripts in order to capture the details of the page within the user's browser window.

Dragging the http links into the Links section of the user's browser will download the tools.

The tools will then pass the information to a JavaServer Page, which will parse the provided information about the page and display this as HTML to allow the user to cut and paste this into their reading list.

Later versions of the system will communicate directly with a VLE to embed the links directly within the appropriate reading list.

2.3 Tool Types

◆ Add link

This tool will simply add the URL and title of the page to the reading list, allowing the user to continue browsing.

◆ Add detailed link

This tool will allow the user to edit information about the page before adding it to the reading list.

◆ Stop

This tool will finalise / close the harvesting window and reading list

2.4 Functional Decomposition

The URL Harvesting Tools development will be broken down into several discreet elements as listed below:

JavaScript development – writing of the various JavaScript tools to pass relevant URL and page title details to the list.

Tool download pages – develop pages to enable download of the various tools into the user's browser.

List Servlet – a servlet will be developed to accept the results from the UHT and create a list of the resources.

This will form the basis of the resource list and will later be incorporated into the required VLE programs.

Controller Servlet – a servlet to allow multiple users to use the system, each being able to log in and create a number of resources lists. This servlet should also allow the user to set an active list to be used in conjunction with the UHT tools.

2.5 Timetable for Development

The initial development will focus on functionality external to any specific institutional systems such as VLEs, MLEs etc.

Early prototypes of the deliverables will be created over the next few weeks, with an alpha system available by the end of March 2003.

Further development will then focus on adapting the functionality to integrate with institutional systems and a beta production version will be produced by the end of June 2003.

A more detailed timetable of development of each module is given below:

Code and Test JavaScript elements – 3 days

Tool download pages – 2 days

Code and Test List Servlet – 3 days

Code and Test Controller Servlet – 8 days