<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEARCHMANAGER Security: Specifying Users/Passwords</td>
<td>34</td>
</tr>
<tr>
<td>Use the SEARCHMANAGER to Configure the Search Environment</td>
<td>35</td>
</tr>
<tr>
<td>Use the SDASEARCH Servlet for Single-Study or Cross-Study Searching</td>
<td>36</td>
</tr>
<tr>
<td>A Note on the Archiver and Searching</td>
<td>36</td>
</tr>
<tr>
<td><strong>APPENDIX A: UPGRADING TO SDA 3.5 FROM AN EARLIER VERSION</strong></td>
<td>38</td>
</tr>
<tr>
<td><strong>APPENDIX B: USING THE TOMCAT SERVLET CONTAINER</strong></td>
<td>40</td>
</tr>
<tr>
<td>Installing and Configuring Tomcat</td>
<td>40</td>
</tr>
<tr>
<td>Connecting Tomcat to Apache or IIS</td>
<td>42</td>
</tr>
<tr>
<td>Apache to Tomcat Connector</td>
<td>43</td>
</tr>
<tr>
<td>IIS to Tomcat Connector</td>
<td>46</td>
</tr>
<tr>
<td><strong>APPENDIX C: AUTHENTICATION AND SPRING SECURITY</strong></td>
<td>49</td>
</tr>
<tr>
<td><strong>APPENDIX D: TROUBLESHOOTING</strong></td>
<td>50</td>
</tr>
<tr>
<td>Debug Mode</td>
<td>50</td>
</tr>
<tr>
<td>IIS and Windows Server</td>
<td>50</td>
</tr>
<tr>
<td>IIS 6 and Windows Server 2003</td>
<td>50</td>
</tr>
<tr>
<td>Some Notes on IIS 7</td>
<td>51</td>
</tr>
<tr>
<td>VarCase Specification</td>
<td>52</td>
</tr>
<tr>
<td>Charts on Linux or Solaris</td>
<td>52</td>
</tr>
</tbody>
</table>
Introduction

What is SDA?

Survey Documentation and Analysis (SDA) is a set of programs for the documentation and Web-based analysis of survey data. SDA is developed and maintained by the Computer-assisted Survey Methods Program (CSM) at the University of California, Berkeley. CSM also develops the CASES software package for collecting survey data.

SDA versions are available for Linux (on Intel x86 and compatible platforms), Windows and Solaris (on the Sparc platform). To set up a basic SDA Web Archive you need to run a Web server that supports CGI -- which includes just about any Web server software. Typically Linux and Solaris users run the Apache Web Server and Windows users run Microsoft's Internet Information Server (IIS) -- although Apache is also available for Windows. (Also, see the warning about the issues with the newest release of IIS -- version 7 -- in "Appendix D: Troubleshooting"). For some additional SDA capabilities -- including charts and searching -- you also need to run a Java servlet "container" such as Apache's Tomcat. The SDA programs are written in C++ (with the exception of the servlet-based programs which are written in Java).

The SDA Web site is at: http://sda.berkeley.edu. The site contains a data archive that demonstrates the various capabilities of SDA, including "real-time" data analysis and creating custom subsets of datasets. It also contains a wealth of SDA documentation, including the full SDA Manual. (From the home page of the SDA Web site click on the "Documentation" link.) The SDA Manual contains full documentation on every SDA program. It also has sections on SDA's Data Description Language (DDL) syntax and on the "HARC" configuration file that is the "nerve center" of an SDA data archive on the Web. Be sure to consult the manual when you need more detailed information about SDA than is contained in this Guide.

The rest of this document contains an overview of the steps to follow in creating a Web data archive with SDA. Here’s the basic “recipe”:

Steps to Follow

1. Install SDA and configure your Web server using the sample dataset:
   - Install the SDA files on your server
   - Configure your Web server for SDA
   - Edit the "harcsda" HARC configuration file
   - Edit the HSDA startup link (if necessary)

2. Set up your own datasets in SDA format:
   - Obtain the ASCII data file
   - Create a DDL file for the dataset
   - Use the ASCII data and DDL file with MAKESDA to create an SDA dataset

3. Create HTML codebooks for your datasets:
   - Create a variable list with headings
   - Create a batch options file for XCODEBK

4. Add your datasets to the Web data archive
   - Add a section in a HARC file for each dataset
   - Add an HSDA startup link for each dataset

Much of the work in steps 2-4 can now be done with the new "Archiver" Web application -- instead of from the command line -- if desired. These steps will be discussed in more detail in the sections below.
A Note on Quick Tables

If your organization is only interested in using SDA's Quick Tables – the simplified interface for creating analysis results – then you do not need to follow all of the steps in this Guide. Specifically, the Quick Tables user interface is powered exclusively by Java servlet technology (not by CGI). Therefore, you do not need to configure your Web server (Apache or IIS) for Quick Tables. Instead, you only need to configure a servlet container (such as Tomcat) and deploy the Quick Tables webapp. Please note however that a number of the sections in this Guide are still relevant to configuring Quick Tables. For example, you will still need to create SDA-format datasets to use with Quick Tables.

Therefore, if you're only interested in setting up Quick Tables here's a suggested reading list – and series of steps to follow -- that omits the irrelevant parts of the SDA Guide:

1) Under the heading "Install SDA and Configure your Web Server" read the first two sections: "Install the SDA Distribution Package on your Server" and "Essential Archive Directories and Files". These sections will get you started with installing the SDA package and becoming familiar with its contents.

2) Read "Appendix B: Using the Tomcat Servlet container". This will get you started setting up and configuring a servlet container for Quick Tables.

3) Read the Quick Tables documentation at: [http://sda.berkeley.edu/info/quicktables](http://sda.berkeley.edu/info/quicktables). Deploy and configure Quick Tables according to the instructions in the documentation. Use the SDA Test Dataset (included in the SDA distribution) to test your Quick Tables configuration. In this Guide, under the heading "Install SDA and Configure your Web Server" read the section "Adding Charts to Tables and Means Output". Quick Tables and the regular SDA interface use the same "chartgen" servlet to generate charts. This section tells you how to configure the chartgen webapp. Once the chartgen servlet is running, your Quick Tables output will include charts.

4) You're now ready to add your own datasets to Quick Tables. Read all the sections under the heading "Files Required to Set Up Your Datasets in SDA Format." Now, under the heading "Archive Your Datasets: the Command Line Method", read the sections "A Prerequisite: On Using SDA Programs from the Command Line" and "Create an SDA Dataset by Combining the DDL File and ASCII Data File (MAKESDA)". If you want to make codebooks, you should also read the next section "Create HTML Codebooks for your Datasets".
Install SDA and Configure Your Web Server

Install the SDA distribution package on your server

The SDA distribution package comes in a single compressed file. Follow the installation instructions below for your platform:

**Linux and Solaris:**

The distribution file has a “.Z” extension and is a compressed tar file. (The main part of the name – before the extension – will vary depending on the version.) Copy the file to the directory where you want to place the various archive directories/files. It's recommended that you name this parent directory "sdaweb" to make your initial installation easier. (The paths in the "harcsda" archive configuration file -- which you'll be editing later -- use this convention so it just makes the procedure a little easier.) Now "cd" into the directory where you've placed the SDA distribution file. To uncompress and “untar” the file follow the instructions in the “readme” file for the distribution.

**Windows:**

The distribution file has a “.zip” extension and is a zip file. (The main part of the name – before the extension – will vary depending on the version.) To unzip it you will need to use a recent version of a compression utility such as WinZip that can handle long file names and will preserve the internal directory structure. If you "extract" the zip file at the root level of your "C:" drive, a "C:\sdaweb" directory should be created which contains all of the various directories and files. You can, however, extract the contents to another drive or directory location on your server -- such as the typical default IIS home directory: "C:\Inetpub\wwwroot".

Once you've installed the package take some time to get familiar with the directory structure and contents of the package. The package contains all the SDA programs and files necessary to set up your own Web data archive with your own datasets. It also contains a sample SDA dataset and HTML codebook to help you get started. Below is a quick overview of the essential directories and files.

A note on the location of directories in the distribution package: you may depart from the layout of the distribution package when you set up an SDA archive on your server. At the very least, the WAR files in the "servlets" directory will necessarily be deployed within your Tomcat installation (or whatever servlet container you use) -- not in the "document tree" or "virtual directory" of your Apache or IIS Web server. As discussed below, you may also want to move the directories that do not need to be "browseable" into locations outside the scope of the Web server. The sections below should make the purpose and contents of the various distribution files clear -- and help you decide where they should go in your archive's layout.

**Essential archive directories and files**

The "index.html" file

The "index.html" file is a simple home page for the sample SDA Web data archive. It contains a hyperlink that invokes HSDA, the top-level SDA CGI program. HSDA launches the whole set of procedures for the Web data archive into action. The "index.html" page also contains a hyperlink that invokes HSDA's "debug mode", which can be very useful in diagnosing problems with your Web data archive configuration.

Once you've got your sample SDA archive up and running you will want to change this file's contents so that the text refers to your own archive material and the links go to your own datasets. (And you'll certainly want to remove the "debug mode" link from the public entry point to your own SDA archive.) In short, this page is just meant to provide simple examples of the start-up link syntax -- both for normal usage and for debug mode.
The “cgi-bin” directory

SDA uses a Web server's CGI (Common Gateway Interface) capability to present dynamic content and process the user's input. The SDA CGI programs are all in the distribution's "cgi-bin" directory. The SDA CGI programs (with the Windows version names in parentheses) are:

- hsda (hsda.exe)
- hsda2 (hsda2.exe)
- hsda3 (hsda3.exe)
- subsdal (subsdal.exe)
- subsda2 (subsdal2.exe)
- subsda3 (subsdal3.exe)

This directory also contains a HARC (HTML ARchive Configuration) file -- named “harcsda” in the distribution package. Every SDA Web Data Archive must have one or more HARC files; most of the information and options for the archive are specified in this file. Later we will describe how to edit the "harcsda" file so you can set up the sample SDA archive on your own server.

The “sda” directory

This directory contains the SDA programs that aren’t CGI programs (and aren't servlets). Some of these programs (such as the analysis programs) are executed directly by the CGI programs. Others (such as DDLMOD and MAKESDA) would only be run by the archive developer while preparing new datasets for the archive. Finally, some of these programs (such as DDLTOX, XCODEBK, RECODE and COMPUTE) would be used both by the CGI programs and by the archive developer. Therefore this directory should be specified both in the user’s PATH environment variable (so the developer can access them) and the HARC file’s SDAPATH (so the CGI programs can access them).

These files are the SDA on-line analysis programs, invoked by SDA’s CGI programs.

- tables (tables.exe)
- means (means.exe)
- corrtab (corrtab.exe)
- correl (correl.exe)
- regress (regress.exe)
- logit (logit.exe)
- listcase (listcase.exe)

These programs are used for online subsetting, invoked by SDA’s CGI program. The archive developer would also use XCODEBK to create HTML codebooks and DDLTOX to create various metadata formats from DDL. SUBSET can also be used as a standalone utility by the archive developer if desired.

- subset (subset.exe) Creates subset ASCII data file and corresponding DDL file from either an SDA dataset or ASCII data file/DDL combination.
- ddltox (ddltox.exe) Converts DDL to SAS, SPSS, STATA or DDI (XML) data definition files.
- xcodebk (xcodebk.exe) Creates codebooks.
Other programs used by the archive developer for creating DDL files, SDA-format datasets, etc.

- **makesda** (makesda.exe) Creates SDA dataset from ASCII data file and DDL file.
- **xconvert** (xconvert.exe) Converts SAS/SPSS/STATA data definition files to DDL or DDI (XML). (Replaces the older XTODDL program.)
- **sdataxml** (sdataxml.exe) Reads an SDA dataset and creates variable definitions in DDI (XML).
- **ddlmod** (ddlmod.exe) Utility program to modify and merge DDL files.
- **q4toddl** (q4toddl.exe) Creates DDL file from CASES 3 or 4 series instruments.
- **qextract** (qextract.exe) Extracts information from CASES instruments, and produces an IDL file.
- **sdalog** (sdalog.exe) Utility program for analyzing the contents of the SDA log file.

Programs for creating derived variables. Can be used by SDA CGI programs and the archive developer:

- **recode** (recode.exe) Recodes variables into a new variable
- **compute** (compute.exe) Performs calculations to create a new variable

Other:

- **describe** (describe.exe) Generates a simple description - usually including a frequency distribution -- for a variable. Invoked with the "View" button in the 3.x user interface.
- **harcrdr** (harcrdr.exe) Utility program used by searchmanager servlet to extract info from HARC files.
- **sdadel** (sdadel.exe) Utility program used internally by SDA CGI programs. Deletes SDA variable files.
- **vartext** (vartext.exe) Utility program used by searchmanager servlet to extract text from VAR files.
- **xtables** (xtables.exe) Extended N-way crosstabulation program. Produces plain text output (not HTML). Useful in special circumstances, but not part of regular SDA analysis package. New in SDA 3.5.

The “helpfiles” directory

This directory contains the online HTML help files for archive users.

- **helpan.htm** - On-line help file for analysis programs
- **helpgo.htm** - On-line "Getting Started" help file
- **helpnewv.htm** - On-line help file for creating new variables
- **helpsub.htm** - On-line help file for subsetting
- **semethod.htm** - On-line help file about methods used by SDA for computing standard errors for complex samples
- **srchhelp.htm** - On-line help file for searching

The "helpfiles" directory also contains CSS files and a "jscript" subdirectory. The "jscript" subdirectory in turn contains various HTML, JavaScript and CSS files that are a necessary part of the SDA user interface. The "jscript" directory
also contains an "icons" subdirectory that contains various graphics files. The contents of the "jscript" directory are too numerous to list here. But be sure to include this directory and all of its contents when you set up your SDA archive.

The "testdata" directory

The "testdata" directory contains the sample SDA dataset and HTML codebook files that you'll use for the initial configuration of your Web server software. The files comprising the sample SDA dataset are in the "STUDYINF" and "VARS" subdirectories. The files comprising the HTML codebook are in the "Doc" subdirectory. The various work files used to create the HTML codebook are in the "Docwork" subdirectory. When you are ready to create your own SDA codebooks the contents of the "Docwork" directory can help you see how the various pieces fit together. Start by looking at "xcodebk.opt", the XCODEBK batch options file that was used to create the sample HTML codebook.

The “testdata” directory also contains the original raw ASCII data file (“sdadat.txt”) and DDL file (“sdaddl.txt”) that were used as input to MAKESDA to create the SDA sample dataset. Also, SDA’s DDLTOX program was used to create the various metadata files based on the DDL file: “sdasas.txt” (SAS format), “sdaspss.txt” (SPSS format), “sdastata.txt” (STATA format). SDATOXML was used to create the DDI (XML) file “sdaddi.xml”. Finally, the directory contains various codebook versions: “cbtagged.txt” (the “tagged” output format used for conversion to Word format), “codebook.doc” (the Word codebook) and “codebook.pdf” (a PDF version created from the Word version). Most of these files are also referenced in the example “harcsda” configuration file to demonstrate the “Download existing dataset and documentation” option.

The “wordmacro” directory

The “wordmacro” directory contains the SDA Word macro file “sdacdbk.txt”. This macro converts SDA codebooks in the “tagged” output format to Word documents. For information on installing and using the macro file see the “WORDMACRO” document in the online SDA Manual.

The "tmpdir" directory

The "tmpdir" directory is an initially empty directory that’s intended to be used for storing various temporary files created by the SDA CGI programs, particularly the creation of various “subset” files. For further information, see the section below entitled “Set up other archive capabilities”.

The “logs” directory

The “logs” directory is an initially empty directory that can be used to store the SDA log file. For more information on this topic see the section on “Logging Archive Usage” below.

The “servlets” directory

The “servlets” directory contains WAR files – and example configuration files – for deployment in a servlet container (such as Tomcat). The directory contains these sub-directories:

The “chartgen” directory

The “chartgen” directory contains the SDA “chartgen.war” servlet WAR file. For more information about how to install ("deploy") this servlet and configure your HARC file for generating charts, see the section “Adding Charts to Tables Output”. The “chartgen” servlet uses JFreeChart, an open source Java library project that uses the LGPL license. To conform to the LGPL requirements, a copy of the license and a copy of the JFreeChart source code are also included. See the “readme.txt” file in the “jfreechart” subdirectory for more information.
The “search” directory

The “search” directory contains the SDA “searchmanager.war” and "sdasearch.war" servlet WAR files. The "searchmanager" web application allows the archivist to configure the archive's search environment from a Web interface. The "sdasearch" servlet provides the support for user searches – both single-study and cross-study. (These search servlets were introduced in SDA 3.2.) The directory also contains a sample search configuration file called "searchconf.txt". For more information, see the section below entitled "Search Configuration".

The “archiver” directory

The “archiver” directory contains the SDA “sda_archiver.war” servlet WAR file. The Archiver web application (introduced in SDA 3.1) provides a way to archive SDA datasets via a browser interface. The directory also contains a sample Archiver configuration file called "appconfig.txt".

The “quicktables” directory

The “quicktables” directory contains the SDA “quicktables.war” servlet WAR file. The "quicktables" web application provides a simplified interface for obtaining analysis results. The directory contains a "readme.txt" file (that should be read first) and an example Quick Tables configuration file that works with the SDA Test dataset.

Configure your web server for SDA

Once you’ve installed the SDA distribution files on your computer you must ensure that your Web server is configured correctly for running an SDA Web data archive:

1. **The SDA files must be in a “document tree” or “virtual directory” on your server**

   This is the most basic step for setting up any Web server: you must map a real directory on your server's drive to a virtual directory name through which it's accessed on the Web. If you've already set up your Web server to serve other content, then you already have at least one “document tree” or “virtual directory”. If you're setting up your server for the first time, consult your Web server documentation for instructions concerning your particular server software.

   Once you've completed this step, check that your SDA archive is Web-accessible by opening the "index.html" page in your browser. If, for example, you've mapped the "sdaweb" directory to the URL

   http://myserver.mycollege.edu/sdaweb

   then you should open the following page in your browser:

   http://myserver.mycollege.edu/sdaweb/index.html

   On the typical Linux/Unix Web server you could omit the final "index.html" from the URL since the server will look for that file in the directory by default. However, if you're using Microsoft's IIS Web server you may need to rename this file "Default.htm" if you want it to be the default file that's opened when you access the "sdaweb" directory.

   Not all the directories and files in the SDA distribution need to be in a document tree or virtual directory – only those that need to be visible or “browseable”. You might want to move some of these directories to other locations on your server for security reasons – or simply to align them with your server's current layout of directories. The directories that can reside outside the Web-accessible area on the server include “cgi-bin”, “sda” and “logs”. Just be aware that if you move these directories (and their files) from their initial place within the SDA distribution package you will have to make the appropriate adjustments in the “harcsda” file.
Don't try to click on the links on the page just yet. You'll need to finish configuring your Web server and editing the "harcsda" file before they'll work correctly.

2. **Configure the "cgi-bin" directory as a CGI (Common Gateway Interface) directory**

This procedure varies for different Web servers. On Linux/Unix Web servers you typically edit a configuration file to specify the location of a CGI directory. (The Apache Web server, for example, uses “ScriptAlias” settings in the httpd.conf configuration file to specify CGI directories.) On Microsoft's IIS Web server you must give the "cgi-bin" directory "Execute" permissions in the property sheet for that virtual directory. (See the section "IIS6/Windows Server 2003" in "Appendix D: Troubleshooting" for more on configuring IIS for SDA.) Also, many servers have a default CGI directory already configured. You can use this default CGI directory or configure one specifically for your SDA archive. Again, consult your Web server's documentation for more detailed instructions.

3. **Create a “\tmp” directory for temporary files (Windows only)**

The SDA programs sometimes need to create temporary files while running. Linux/Solaris systems generally have a built-in “\tmp” directory that SDA uses as a place to store these files. However, if you’re running on a Windows system you’ll need to manually create a “\tmp” directory on the drive containing the SDA CGI programs. The only requirements are: 1) this directory is at the “root” level ( “\tmp” is correct but NOT “\mystuff\tmp”); 2) the directory is writeable by the CGI process. (Note that, unlike the “SUBTMPDIR” directory, this directory does not need to be within a “virtual directory” or “document tree”.) Alternatively, you can define either a “TMP” or “TEMP” environment variable that points to a different directory. (Consult your online Windows help for information on creating environment variables.) Please note that this directory for temporary files must be created and/or specified even if you’ve also specified a “SUBTMPDIR” directory in your HARC file (for use with subsetting).

**Edit the "harcsda" HARC configuration file**

The "harcsda" HARC configuration file is a plain text file that specifies information and various options for your SDA Web data archive. The file is initially placed in your “cgi-bin” directory (although it can be placed elsewhere). The layout and contents of this file are described in detail in the “HARC” section of the online SDA Manual.

Note that a large SDA data archive will often use multiple HARC files. You might organize your datasets so that different groups, with different options, are specified in different HARC files. You might use another HARC file for testing purposes so you can try out new options without disturbing your public production archive. No matter how many HARC files you have, or how you name them, you just need to specify the appropriate one when you create the HSDA start-up link for a dataset. (The syntax of the HSDA start-up link will be discussed later in this document.)

Here’s an example of a simple HARC file (with comments):

**Example HARC Configuration File**

```plaintext
# Comments start with a '#' in the first column.

[GENERAL]

# Enable debug mode
DEBUG = YES

# URL for the directory containing various help files
# (and the jscript subdirectory)
HELPDOCS = http://myserver.mycollege.edu/sdaweb/helpfiles

# Full path for the SDA log file.
LOGFILE = /mycomputer/sdaweb/logs/sdalog
```
The SDA distribution includes a sample SDA dataset and matching HTML codebook. You should now edit the "harcsda" configuration file so these sample archive files work on your system. The editing should be fairly minor: the "place-holder" directory and URL pathnames just need to be modified to match the actual directory and URL pathnames on your server. The supplied "harcsda" file contains extensive comments to help you do this. Once you've edited the path and URL names in the "harcsda" file, re-open the "index.html" file (or "Default.htm" if you've renamed it) and click on the link to the "SDA Test Dataset". If this HSDA start-up link is correct for your server you should now see an options selection form with choices for browsing the codebook and running various analysis programs. If you get an error message instead, the following section will help you fix the start-up link.
**Edit the HSDA start-up link (if necessary)**

If the default "SDA Test Dataset" link on the "index.html" page doesn't work correctly on your system, here's some additional information about what might be wrong and how to fix it.

First, view the HTML source for the index.html page. The HSDA start-up link will look something like this:

(On Windows:)

```html
<a href="cgi-bin/hsda.exe?cgi-bin/harcsda+sdatest"> SDA Test Dataset </a>
```

(On Linux/Solaris:)

```html
<a href="cgi-bin/hsda?harcsda+sdatest"> SDA Test Dataset </a>
```

Note that the "href" attribute is of the general form:

```
[hsda_CGI_program]?[harcfilename]+[datasetname]
```

When the user clicks on the link, the **HSDA** program (in “cgi-bin”) is executed. **HSDA** is passed the “HARC” file name (“harcsda”) and the dataset name (“sdatest”). **HSDA** reads the “harcsda” file to store the information needed to display the various HTML forms and process user input.

If the link isn't working the likely problem is either that the HSDA CGI program can't be executed by your Web server or the HARC file can't be found.

If the CGI program ("hsda" or "hsda.exe") can't be executed then you'll get an error message from your Web server or browser (rather than an error message generated by SDA). Check that the name of the program is correct -- "hsda.exe" if you're on Windows or just "hsda" otherwise. Also, check the documentation for your Web server – Apache or IIS – to make sure you've taken the necessary steps in configuring cgi-bin as a CGI directory.

Once the server can find (and execute) the HSDA CGI program the link should work correctly. However, if HSDA executes, but reports that it can't find/open the "harcsda" file, you need to change the path to the "harcsda" file. This is usually only a problem on Windows. When a CGI program runs in a standard, non-IIS Web server, the "current working directory" for the CGI program is usually the CGI directory itself. Therefore, if the "harcsda" file is in the same directory as the CGI program that opens it, you only need to refer to the file's name. However, Microsoft's IIS treats the "current working directory" of a CGI program a bit differently: the working directory of a CGI program is at the root of the virtual path that includes the CGI directory. So if the "sdaweb" directory is mapped as a virtual path, and the "cgi-bin" directory is a child of that virtual path, then the "current working directory" of a CGI program in "cgi-bin" is "sdaweb" (not "cgi-bin"). Therefore, using IIS, the reference to the "harcsda" file must include the "cgi-bin" sub-directory information: "cgi-bin/harcsda". The supplied "index.html" file uses this syntax. However, you may encounter problems if you placed your SDA archive in a directory other than "sdaweb". If so, edit the link's "harcsda" path information to match your own configuration. (Note that if you're using a non-IIS Web server on Windows -- such as Apache for Win32 -- you may need to remove the "cgi-bin/" path information in front of "harcsda" since the Apache server treats the CGI "current working directory" like a typical Unix Web server.) Finally, if all else fails, you can simply give the full, absolute directory path to the "harcsda" file. For example:

```java
"cgi-bin/hsda.exe?C:\sdaweb\cgi-bin\harcsda+sdatest"
```

Once you can get HSDA to find and read the HARC file, you can use HSDA's debug mode (see below) to determine the "current working directory" of the HSDA program. (The "current working directory" of the CGI process is listed in the debug mode output.) With this information you can then abbreviate the path name to the HARC file if you wish.
**Use HSDA in “debug mode” to check for errors in your archive configuration**

The default “home page” for the example SDA archive (the “index.html” file) has a “Debug mode” link. Click on it to check for errors in your archive configuration. (Note that if you had to make changes to the “SDA Test Dataset” link, you will need to make similar changes to the “Debug mode” link.)

Debug mode can be very useful in diagnosing problems with an archive's configuration – both during initial setup and later when you make additional modifications to your HARC file(s). Invoking debug mode is straightforward: if HSDA is executed with the word 'debug' substituted for a dataset name, HSDA will check the HARC file for various errors and send a report to the browser. For example, the following hyperlink on an HTML page might be used to invoke debug mode:

```
<a href="cgi-bin/hsda?harcsda+debug"> Debug mode </a>
```

Debug mode for a particular HARC file is disabled by default. To enable it, insert a “DEBUG = YES” line at the beginning of the “[GENERAL]” section of the HARC file you wish to debug. Once the HARC file is checked and debugged, debug mode can be disabled by removing the “DEBUG = YES” line or commenting it out with a ‘#’ in the first column.

**Test Your SDA Archive**

Once your HSDA start-up link is working and debug mode shows no errors in your HARC file, browse the example HTML codebook and run the analysis programs using the example SDA dataset. Also, try the option to "Download existing dataset and documentation". (If you run into problems, check Appendix D: Troubleshooting.) Once these basic archive capabilities are working you can continue by setting up the additional SDA archive capabilities listed in the next section. Or you can skip forward to “Set Up Your Datasets in SDA Format” and start creating your own SDA datasets and HTML codebooks and add them to your archive. Then you can come back here when you’re ready to set up logging, customized subsetting, the creation of new variables and the creation of charts.

**Set up other archive capabilities**

One thing to note about the capabilities listed below: they all involve procedures where the CGI process is writing one or more files to disk. Logging creates an initial log file then appends records to it. Subsetting writes various files to a browser-accessible directory for downloading. Variables created by **RECODE** and **COMPUTE** are stored in an SDA “VARS” directory. Generating charts depends on the **TABLES** program writing temporary chart specification files that are then read by a servlet that generates the actual chart image. In each case, the directory where these files are written must have its permissions set correctly so that the CGI process has write access. On a Linux or Solaris system, for example, this will often mean that these directories must be writeable by "the group" – or even “the world” – depending on how your Web server is configured. On a Windows IIS server you may have to explicitly allow the "Internet Guest Account" to write files in the necessary directories. In short, enabling the following capabilities may require a little tweaking of the permissions or security settings on your system.

**Note: adding search capabilities to your archive is covered in the section below entitled "Search Configuration".**

**Logging Archive Usage**

Before you make your SDA archive public you should ensure that SDA's logging mechanism is working. The "SDALOG" specification in the HARC file determines the name and location of the SDA log file. A record is written to this log file by one of the SDA CGI programs each time a user runs an analysis or creates a custom subset, which is very handy for determining how your archive is actually being used. The SDA distribution package also contains an **SDALOG** program that creates various reports based on the contents of this log file. Note that the SDA log file is a completely different file -- with a completely different format -- than the log maintained by the Web server. The
server's log either tells you too much or too little about site usage. For example, the server's log can't tell you how many times your users ran TABLES or MEANS. (It just tells you that a CGI program like HSDA was invoked.) And since the server's log contains a record for every http request (including the download of images and requests made by search "robots"), it is difficult to see how people are really using your Web data archive.

Customized Subsetting

The "harcsda" file contains some information and example specifications to help you configure the customized subsetting utility. Subsetting obviously requires a place to store the files users create. The SDA distribution package contains a “tmpdir” directory that can be used for this purpose. (Again, this directory must be writeable by the process that is running the CGI programs.) Also, you must specify the location of this directory for temporary subset files using the “SUBTMPDIR” keyword in your HARC file. Note also that this directory must be within a “virtual directory” or “document tree” if it will be used for subsetting. So you must also specify the associated URL for this same directory using the “SUBTMPURL” keyword. Therefore, the [GENERAL] section of the HARC file must contain specifications of the following form:

SUBTMPDIR = [full path of directory, e.g. “/mycomputer/sdaweb/tmpdir”]
SUBTMPURL = [full URL of directory, e.g., “http://myurl/sdaweb/tmpdir”]

There are also specifications to consider at the [DATASETS] level of the HARC file. First, for each dataset that will allow customized subsetting you should specify the following:

SUBGRPINFO = [full pathname of file with info on groups of variables]

The SUBGRPINFO file is a special file that is generated automatically whenever the XCODEBK program produces HTML codebook files. It has the name “Xsub.txt”, where “X” is the root name of the HTML codebook files. It is output to the same directory as the codebook HTML files. Although specifying the SUBGRPINFO file is not strictly required, it is highly recommended as it enables the user to select entire groups of variables (based on the group headings in the XCODEBK varlist), instead of having to specify all desired variables one by one.

What if you don't want to allow subsetting for a particular dataset listed in your HARC file? Within the specifications for that dataset you simply specify:

SUBSET = no

(Also, if you want to be absolutely sure that the subset procedure is not available for a particular dataset – for confidentiality or other reasons -- you should set up a disclosure file for that dataset and include the specification "subset = no" in that disclosure file. See the "Disclosure" document in the online SDA Manual for more information.)

Since SDA 3.3 subsetting has – by default -- used the variables in an SDA dataset as the input source. This allows variables created by RECODE or COMPUTE to be included in customized subsets. If you're using this default subset mode, then you do not need to specify anything more. However, if you want to produce subsets the old-fashioned way -- or you're using a pre-3.3 version of SDA -- then you'll need to specify the source raw data and DDL files in the [DATASETS] section:

SUBDDL = [full pathname of DDL file]
SUBDATA = [full pathname of ASCII data file]

The SUBDDL and SUBDATA files would normally be the same files that you use with MAKESDA to create the SDA dataset. Also, unless you're using a pre-3.3 version of SDA you'll need to put the following specification in the [GENERAL] section of the HARC file:

OLDSUBSET = yes
This specification will cause subsets to be produced from the specified raw data and DDL files. However, variables created by RECODE or COMPUTE cannot be included in subsets if this mode is used.

Finally, you probably will want to run some program that automatically cleans out the SUBTMPDIR directory periodically.

Allowing Creation of New Variables

You can also configure your SDA archive so that users can create new variables using RECODE and COMPUTE. This topic can be complex and a full discussion is beyond the range of this Guide. (For more details see the sections on the HARC file and HSDA in the online SDA Manual.) But we’ll briefly outline the main ideas here.

The first step in allowing the creation of new variables is to add the appropriate SDA programs to the [PROGRAMS] section of the HARC file. Normally you’d add “recode”, “compute”, and “listvars(delete)” to the programs specified by the “SDAPROGS” keyword. The “listvars” program doesn’t create new variables, but it allows users to see a listing of the new variables that have been created. The “(delete)” option after the program name enables the user to delete variables in the list.

The second required step is to specify the name of the directory in which to store the new variables. There are a couple of ways to do this, but the most straightforward is to put an “OUTSTUDY=[directory]” keyword into the HARC file at the dataset level.

It isn’t enough to just allow the creation of new variables. Users will want to include the new variables they’ve created in their analyses. To do that, you also need to create an “SDADATA = [directory]” specification for the directory that stores the new variables. So the “SDADATA” and “OUTSTUDY” specifications for a dataset that allows the creation of new variables will typically look like the following example:

```
# Specification of original SDA dataset directory for the study
SDADATA = /mycomputer/studies/gss2002
# Specification of directory for new variables
OUTSTUDY = /mycomputer/newvars/gss2002
SDADATA = /mycomputer/newvars/gss2002
```

One other task remains to be done: the directory that stores new variables needs to contain the “skeleton” structure of an SDA dataset before new variables can be added. So you need to create both “STUDYINF” and “VARS” subdirectories in the OUTSTUDY directory. Then copy the CASEID variable file (from the original SDA dataset) into the new VARS directory. Also, copy the “studyinf” file (from the original SDA dataset) into the new STUDYINF directory. (Finally, again, the VARS directory of the OUTSTUDY must be writeable by the process that is running the CGI programs.)

This method of specifying the “OUTSTUDY” directory creates a public work area which is available to all users of the dataset. All user-created variables go into the same directory and can be used by anyone else – but also overwritten or deleted by anyone else. An alternative method of enabling the creation of new variables is to invoke the HSDA program with the name of an “OUTSTUDY” given as an argument. Different users can then share the main dataset for a study but store new variables in their own private work area. This configuration is somewhat complicated and labor-intensive to set up. But it can be useful in some limited situations. For more information see the online HSDA documentation.
Adding Charts to Tables and Means Output

SDA 1.4 added the capability of generating charts in the output of the TABLES program. SDA 3.4 extended this capability to MEANS. This chart generation capability uses Java servlets technology. That means you'll need to run a servlet "container" or "engine" on your server to generate charts. (Of course, this is entirely optional. If, for some reason, you don't want to generate charts on your site, you can run the rest of the SDA Web data archive software using just CGI as before.) If you aren’t already running a servlet container, see Appendix B of this document – “Using the Tomcat Servlet Container” – for more information about obtaining and using this free, open-source and industrial-strength servlet container software.

Once you have a servlet container running on your server, setting up chart generation is quite straightforward. First, install (or “deploy”) the SDA chart generation servlet in your servlet container. The SDA chart generation servlet is packaged as a standard "WAR" (Web ARchive) file called “chartgen.war”. To deploy it in your servlet container you can just follow the procedure you use for any other "WAR" file. Most simply, on Tomcat, you can often just copy the chartgen.war file into the Tomcat “webapps” directory. (Depending on your Tomcat configuration, you may also need to restart Tomcat.) Or you can use the Tomcat Manager to deploy the chartgen.war file.

Once the “chartgen.war” servlet is deployed, you’ll need to make some changes to your HARC file. (All of these chart specifications should go in the [GENERAL] section of the HARC file.) First, using the “CHARTGENURL” keyword, you must specify the URL where the deployed chart generation servlet can be accessed. If, for example, you’re using Tomcat’s default port of 8080 and your server’s name is “myserver.mycollege.edu”, then the specification would look something like this:

CHARTGENURL = http://myserver.mycollege.edu:8080/chartgen

(Of course, the URL of the chart generation servlet -- including the port -- will vary depending on how your servlet container and Web server are configured. It is recommended that you use a "connector" between your servlet container and your Web server so you don't need to specify a separate port number. See the section below in Appendix B on "Connecting Tomcat to Apache or IIS" for more information.)

You should also specify the full directory path where the temporary chart specification files will be created by the TABLES or MEANS program and then read by the chartgen servlet. Use the “CHARTSTMPDIR” keyword for this. For example:

CHARTSTMPDIR = /mycomputer/chartstmp

To explain this a bit: when the analysis program executes (and a chart is requested) it will create, in addition to the regular HTML output, a temporary chart specifications file (with a “.cht” suffix) that contains all of the information (in a plain text format) needed to create an appropriate chart image. In addition, within the HTML output, the analysis program inserts an image tag something like the following (slightly simplified) example:

```html
```

Note here that the image “src” (source) attribute points to the URL of the chartgen servlet which you specified with the CHARTGENURL keyword. Also, the source information includes the location of the chart specification file where the image properties can be found. The directory where this temporary file is created is the one you specified with the CHARTSTMPDIR keyword. The chartgen servlet then reads this file of chart specifications and creates the chart image – which is then displayed by the browser. All of this has a few implications for the CHARTSTMPDIR directory: it must be writeable by the analysis program and at least readable by the chartgen servlet. However, it is somewhat advantageous if the CHARTSTMPDIR directory is also writeable by the chartgen servlet, since the servlet will try to delete the temporary chart specifications file once it has created the chart image. (This is just a bit of helpful housekeeping done by the chartgen servlet.)
If you do not specify a CHARTSTMPDIR directory, but you’ve specified a SUBTMPDIR directory for subsetting, then the SUBTMPDIR directory will be used for the temporary chart specification files instead. If neither a CHARTSTMPDIR nor a SUBTMPDIR directory is specified, then the system’s default temporary directory will be used. However, it’s probably best in most circumstances to specify an explicit CHARTSTMPDIR directory if you’re using SDA’s charting capability. (Note also: the CHARTSTMPDIR directory does not need to be browseable. That is, unlike the SUBTMPDIR directory, it does not need to be within a virtual directory or document tree.) Like the SUBTMPDIR directory, you will want to run some program that automatically cleans out the CHARTSTMPDIR directory periodically. Even if the chartgen servlet is usually able to delete the chart specification files it has already processed, various scenarios can occur that will cause “.cht” files to be created but not automatically erased.

MAXCHARTS is another, optional, specification that can be placed in the [GENERAL] section of the HARC file. MAXCHARTS simply specifies the maximum number of charts that will be generated per each invocation of the TABLES or MEANS program. Just as multiple HTML tables are generated if a control variable or multiple row and/or column variables are specified, multiple chart images will be generated as well. However, since generating chart images is usually relatively expensive in terms of resource usage compared to generating HTML tables, there is a default maximum of 25 charts created per analysis run. However, this maximum can be redefined using the MAXCHARTS keyword. For example:

MAXCHARTS = 15

This will set the maximum to 15 (instead of the default 25). Any integer from 1 to 100 is legal. However, you should be careful about setting the maximum higher than the default 25 since users can easily request dozens of charts in a single analysis run – perhaps inadvertently. Setting MAXCHARTS too high could put an unnecessary strain on your server’s resources – and your user’s patience. On the other hand, you might want to set MAXCHARTS to less than the default if you find it is too demanding on your server’s memory and CPU.

Finally, the optional TABLESCHARTS and MEANSCARTHS keywords allow you to limit the types of charts available, change the order of the charts on the drop-down menu, or change the label for each type of chart. For more on this subject, see the HARC file documentation in the online SDA Manual.

Modifying the default font and/or color palette for charts:

The chartgen webapp was enhanced in SDA 3.4 so the chart font and color palette can be specified by the archivist in configuration files. If you want to customize your chartgen webapp in this way, you’ll need to take the following steps.

1) Edit chartgen’s web.xml file. Look for the "chartconf" context-param element. By default this element is commented out so you’ll need to remove the "<!––" and "––>" comment tags that surround it. Now look for the <param-value> element that specifies the location of the "chartconf.txt" file. Edit this path so that it matches the location of the chartconf.txt file that you'll create in the next step. The chartconf.txt file can be placed anywhere on your system – so long as it is readable by the chartgen webapp. (It does not need to be in a virtual directory or document root.)

    <context-param>
        <param-name>chartconf</param-name>
        <param-value>c:/chartgen/chartconf.txt</param-value>
        <description>Location of chart generator configuration file</description>
    </context-param>

2) Create a chartconf.txt file. The chartconf.txt file is a plain text file with a very simple syntax. Here's an example:

    # Lines beginning with a '#' are treated as comments
    fontname = Arial
    barcolors = c:/chartgen/barcolors.txt
    linecolors = c:/chartgen/linecolors.txt
All of the possible specifications – "fontname", "barcolors" and "linecolors" – are optional. You can use any or all of them.

The "fontname" specification specifies the font name that should be used for creating the labels, legends, etc. on your charts. The font name used by default (as of SDA 3.4) is "SansSerif" – one of the five Java "logical" font families. These logical fonts are mapped to physical fonts by the Java runtime environment – so the look of these fonts may differ on various systems. If you want to specify an alternative font, just specify the name here. Sometimes specifying a font is just an aesthetic preference. However, in some circumstances you may need to specify a certain font so that a particular language can be displayed properly.

The "barcolors" and "linecolors" keywords are used to specify the location of files that specify the color palette(s) for your charts. The barcolors specification is used for bar charts and pie charts. The linecolors specification is used only for line charts. Why allow the specification of two separate color palettes? Line colors typically need to be fairly dark to be easily visible. The colors used in bar and pie charts can be somewhat lighter if desired. (And remember not to make bar chart colors too dark – or the percents printed in black on top of the bars might not be legible.)

The files that specify color palettes also have a simple syntax. Here's a (truncated) example with comments:

```
# (Note: all content after a '# ' character on a line is treated as a comment.)
# Each line in the list below contains the RGB values for a color (separated # by commas).
# RGB values can alternatively be expressed in hex. That is, "255,64,64" could # instead be written as "0xFF,0x40,0x40". Remember that valid RGB values # must be between 0-255 (inclusive).
255,64,64
128,128,255
0,255,0
255,255,0
255,200,0
255,0,255
0,255,255
255,175,175
128,128,128
192,0,0
# ...remaining colors are not shown for brevity's sake
```

The colors specified in a color palette file will be used to draw the chart colors in the order they appear in the file. How many colors need to be specified? Although most charts use only a few colors, some charts can include many categories – and therefore many colors. You should probably specify about 30 colors. If a chart needs more colors than are specified in the color palette, the existing colors will be "recycled" as needed.

Once you've finished creating your chartconf.txt file (and any accompanying color palette files) you will need to restart the chartgen webapp for the changes to take effect.

**Modifying the label text for charts:**

The default text of a chart's labels can be changed by specifying the preferred text in the "langan" language file. See the "Interface" document in the online SDA Manual for more information on the "langan" file.

**A Note on Chartgen Versions:**
Please note that pre-3.4 versions of chartgen are incompatible with SDA 3.4 and later. If you are upgrading to SDA 3.5, you must be running at least the 3.4 version of the chartgen webapp.

A Note on Chartgen and Quick Tables:

Quick Tables also uses the chartgen webapp to include charts in its analysis output. If you've already deployed and configured chartgen for the standard SDA interface, then you can just specify the same chartgen URL in your Quick Tables web.xml file.

Other SDA Capabilities

Finally, it's worth mentioning a few other SDA capabilities and where you can find more information about them.

- **Word codebooks**: Besides HTML (and simple text) codebooks, SDA can also produce Word codebooks. For information about how to do this, see the "Wordmacro" section in the online SDA Manual.

- **Complex Standard Errors**: For information on SDA's capabilities for producing standard errors for complex samples, see the "Stderrs" section in the online SDA Manual.

- **Language**: For information on modifying the language of the SDA user interface and analysis output, see the "Language" and "Interface" sections in the online SDA Manual.

- **Disclosure specifications**: To help protect confidentiality, disclosure specifications can be added to an SDA dataset. For information about how to do this, see the "Disclosure" section in the online SDA Manual.
Files Required to Set Up Your Datasets in SDA Format

Now that you've installed SDA, configured your Web server and successfully set up the sample SDA dataset, you're ready to add your own datasets to your SDA archive. For each dataset you will add to your SDA archive you need two files: 1) an ASCII data file; 2) an SDA DDL (Data Description Language) metadata file. In addition, the creation of a variable list with headings is highly recommended. Before talking about how to use these files with SDA, we'll talk a little more about the files themselves.

The ASCII data file

You need to obtain or generate an ASCII data file for each dataset. This file must have each variable located in a fixed set of columns. (“Free format” files, such as comma-delimited or “CSV” files, cannot be used to create an SDA dataset.) Once you have a fixed-format ASCII data file for a dataset, it is necessary to describe that data file using SDA’s Data Description Language (DDL).

The Data Description Language (DDL) file

What is a DDL file?

Data Description Language (DDL) is used for describing the characteristics of a dataset. (Such files are sometimes called "metadata" files.) The descriptions are of two types: a description of the study as a whole, and a description of each variable.

The description of the study as a whole includes a title for the dataset, the pathname of the directory into which the SDA dataset should be placed (by the MAKESDA program), the number of records per case in the data file, the length of each record (number of characters), and (optionally) the number of cases in the data file. Default values for many of the characteristics of individual variables may also be specified there.

The description of each variable MUST include its name, its location in the data file (the column, and also the record number and width if different from the default), and the number of decimal places (if other than the default number of implied decimal places).

Each variable description MAY also include a long label, descriptive text (such as questionnaire wording), and labels for code categories. If some of the code values represent invalid response codes, they may be flagged for exclusion from analysis; a minimum and a maximum valid code can also be specified.

DDL files have a relatively simple syntax and are meant to be easy to read. Here's a brief snippet of the first part of a typical DDL file:

Example DDL File

```
title = Election Study
records/case = 10
reclen = 80
path = mystudy
*
name = CASEID
label = Case ID of Respondent
record = 1
column = 1
```
Ways to create a DDL file

There are various ways to create a DDL file:

1) If you have a data description file in the format of the Data Documentation Initiative (DDI) you can use SDA’s online “DDI to DDL Conversion Service” to convert the DDI file to the DDL format. A link to the conversion service is available from the “Documentation” page on the SDA Web site. (For more information about DDI see the organization’s Web site at: http://www.icpsr.umich.edu/ddi.)

2) If you have an SPSS system file or portable file, you can use the MAKEDDL.SPS SPSS-Python script (provided courtesy of ICPSR). The script converts the SPSS file into an ASCII data file and matching DDL file. The script -- and documentation on its use -- is available on the "Community" page at the SDA Web site.

3) Convert SAS, SPSS, or Stata data definitions into DDL by using the SDA program XCONVERT. Each variable defined for one of those systems will produce a DDL segment that includes the variable name, the variable label, the category labels, and missing-data information. The locations of the variables must be given as a fixed set of columns; freefield entry and format statements are not recognized. Note that the data definitions for those systems do not include the text of the questions asked in a survey. In order to add question text to the DDL file produced by XCONVERT, you can do one of two things:

   Add the question text manually into the DDL file, or

   Create a separate DDL file with only the question text and the names of the items and then merge that information into the main DDL file using the DDLMOD program. (See the discussion of DDLMOD below.)

4) If you’re a CASES 5 series user, the new version of CASES includes a utility that produces DDL directly. If you’re a CASES 4 (or 3) series user you can extract the DDL information from a CASES instrument by using the SDA program Q4TODDL.

5) Or you can simply enter the specifications directly into a file using any ASCII text editor. If you have a relatively small number of variables, this is a reasonable thing to do.
Modifying and combining DDL files (DDLMOD)

**DDLMOD** reads in one DDL file, and then reads in a second DDL file with additional or corrected specifications. It produces a new DDL file containing the additions or modifications. Ordinarily the second file is another DDL file used only to supply DDL specifications **lacking** in the first file, such as `text=' specifications for certain variables, or definitions for new variables. Any specifications already present in the first file, such as labels or missing-data codes, are not changed. For example, if you want to add new material in ‘DDL2’ to ‘DDL1’ – with the modified DDL file written to ‘new ddl’ – then you would use the command line:

```
ddlmod -i DDL1 -j DDL2 -o newddl
```

This usage does not replace any specifications already contained in ‘DDL1’. However, if the `-m' option is used, the contents of the second file will modify any definitions already present in the first file:

```
ddlmod -m -i DDL1 -j DDL2 -o newddl
```

**DDLMOD** can also be used to modify a single DDL file in various ways. For example, it has options to turn all variable names in a DDL file into upper or lower case. Consult the online SDA Manual for more information.

**The Variable List with Headings**

Strictly speaking, a variable list with headings is not required when archiving a dataset. However, for any study with more than a few variables, headings are, in fact, a practical necessity in creating a user-friendly archive. Headings are used to create both the index of headings in the codebook and the tree menu structure in the user interface. Headings make variables easier to find and indicate the natural groupings of items in a study. We highly recommend that you create a variable list with headings for any dataset with a significant number of variables.

You don't need to create a variable list from scratch. Each time **MAKESDA** is invoked it generates a list of all the variables created. You can just insert headings into this list. The construction of a variable list with headings is discussed below in the section "Insert headings into the variable list generated by MAKESDA".
Archive Your Datasets: The Command Line Method

Before SDA 3.1, adding datasets to an SDA archive had to be done from the command line. This method of archiving datasets is often quite efficient – especially since the SDA command line programs can be easily invoked via various scripting languages. Organizations that must process large numbers of datasets can combine SDA programs in scripts so that batches of datasets can be processed in an almost fully-automated fashion.

However, in other situations a more interactive approach – using a graphical user interface – is more useful. Therefore SDA 3.1 introduced an "Archiver" Web application that allows you to add datasets to your SDA from a browser. In this section we will discuss the traditional command line method for archiving datasets. In the next section we will discuss the Web-based Archiver.

**A prerequisite: on using SDA programs from the command line**

The SDA programs for creating SDA-format datasets, producing codebooks, etc., are command-line programs. For Linux or Solaris users this is a familiar way to run programs. For some Windows users, however, this may be a bit foreign. If you're one of the latter, here's a little information to put you at ease.

First, to run a command-line program in Windows you need to start up a commands window, called a “Command Prompt” or "Console Window" depending on the version of Windows you have. (Or, colloquially, it’s sometimes just called a “DOS box”.) For example, on Windows 7, Vista or XP you click on “Start”, then “All Programs”, then point to “Accessories” and click on “Command Prompt”. Once you have a commands window you can enter the SDA program name (and options) at the command prompt -- which usually looks something like "C:\>". However, you'll also need to specify the directory containing the SDA command-line programs in the "PATH" environment variable so that the operating system will be able to find each SDA program when you run it. Again, there are various ways to set the "PATH" on Windows. On Windows XP, for example, you can change environment variables by opening “System” in the Control Panel, selecting the “Advanced” tab, and then clicking on “Environment Variables”. For other Windows versions, search for "environment variable" in the online help.

The SDA command-line programs get their options in one of two ways. If the number of options is small enough, you typically just specify the options by using some simple "flags" at the command line. For example, to create an SDA dataset with the **MAKESDA** program you would start it with a command like:

```
makesda -l myddlfile.txt -d mydatafile.txt
```

The "-l" and "-d" are command line "flags" that indicate the next "argument" (or word) on the command line will specify a certain type of option: "-l" is followed by the DDL file name and "-d" is followed by the ASCII data file name.

SDA programs with more numerous and/or more complicated options use a "batch" file to specify options. For these programs you create a batch commands file -- using a plain-text editor such as "Notepad" -- then just specify the batch file name on the command line. For example, you run the **XCODEBK** program with a command line like:

```
xcodebk -b mybatchfile.txt
```

The substantive options are actually contained in the file "mybatchfile.txt".

For each SDA program, the method of specifying program options and the option syntax is discussed in detail in the online SDA Manual (available in the "Documentation" section of the SDA Web site).
**Create an SDA dataset by combining the DDL file and ASCII data file (MAKESDA)**

Once you have an ASCII data file and a DDL file that describes it, you convert the data into an SDA dataset by running the MAKESDA program. Other SDA programs can then use that dataset.

Running MAKESDA is very simple: you just specify the names of the data and matching DDL files at the command line. For example: makesda -l myddl -d mydata

(Note that the '-l' flag is a lower-case 'L'.)

MAKESDA has various options. It can create an entirely new dataset, or add new variables to an existing dataset, or modify (overwrite) existing variables. (See the online SDA Manual for a full discussion.) If variables are added to an existing SDA dataset, MAKESDA checks the contents of the CASEID variable to make sure that the CASEID value for each case matches the value stored previously in the SDA dataset. It also checks the contents of CASEID if variables are being modified.

A list of the variables defined in the DDL file is written into the file 'MAKESDA.LST' whenever MAKESDA is run. If that file already exists (from a previous MAKESDA run), it is overwritten. This list of variables can be very useful for creating a variable list file for the XCODEBK program.

The data file used as input to MAKESDA must be a standard ASCII file, having a fixed number of records for each case. If a record is shorter than the number of characters defined by the `reclen=` or `lrecl=` keyword, it is padded at the end with blanks.

**Structure of an SDA Dataset**

The structure of an SDA dataset is very simple. Each SDA dataset contains a “STUDYINF” subdirectory and a “VARS” subdirectory. The “STUDYINF” directory contains study-level information of various kinds (including search indexes if they have been created with the SearchManager). The “VARS” directory contains one file for each variable in the dataset; the file’s name matches the variable name. The files in the “VARS” directory are binary files (NOT text files) so they cannot be read or updated with a text editor. If you need to re-create a variable in an SDA dataset because the data or DDL information has changed, just re-run MAKESDA using the updated input files. If you’re changing all or most of the variables in your dataset, just use the “-m” option to overwrite all the existing variables. If you have a large dataset and you’re just changing a few variables, you can manually remove those particular variable files in the “VARS” directory then re-run MAKESDA without the “-m” option. (This will just save the time which would be otherwise required to re-write the unchanged variables.)

*If you use multiple operating systems you should be aware that the variable files in the “VARS” directory have the same format on all platforms.* That means you can transfer these files from one system to another – using binary mode “ftp” or an equivalent – and they will still work the same.
Create HTML Codebooks for Your Datasets

For each of the SDA datasets you have created, you need to generate an HTML codebook to describe that dataset. The XCODEBK program creates these HTML codebooks automatically. But before running that program, you need to create a variable list (with headings) and a batch commands file.

It is a good idea to put all the work files used to create the documentation in a separate directory named, for instance, 'Docwork'. The HTML codebook files should also be placed in a separate directory, here named 'Doc'. These two directories are usually created as subdirectories within the main directory for a dataset.

Insert headings into the variable list generated by MAKESDA

MAKESDA always generates a variable list in the “MAKESDA.LST” file when you run it. You should now add headings to a copy of this list of variables. (Use a copy so that your edited list isn't inadvertently overwritten if you re-run MAKESDA.) The headings in the variable list are extremely important in creating a user-friendly archive. They are used to create both the index of headings in the codebook and the tree menu structure in the user interface. For any study with more than a few variables, headings are, in fact, a practical requirement. And headings are very simple to create. A heading inserted into the list of variables is usually just a line with "**" at the beginning, followed by the text of the heading. Sub-headings (2nd-level headings) can be created by just inserting a "2" in front of the asterisks. (See the section on "symbols used in variable lists" in the summary of XCODEBK functions and formats for more details). Here’s an example of a variable list with headings for a small study:

Example Variable List with Two Levels of Headings

** CASE IDENTIFICATION
CASEID

** SURVEY VARIABLES
2** Government Spending
spend
spend2
spend3
spend4
2** Political Ideology and Party
ideo
party
2** Background Variables
age
educ
gender
race
marital

** WEIGHT VARIABLE
casewt

As we mentioned, the headings in the list structure the main index in the codebook. Just as important, they structure the tree menu in the new user interface. The variable list above will produce the following tree menu in the SDA user interface:
When you create the batch commands file for XCODEBK, just specify the name of the file containing the variable list with headings after the 'varlist' keyword.

Prepare a batch commands file

The command file for the XCODEBK program specifies the name of the file containing the list of variables (and headings) and also the other options you want. The XCODEBK program has a large number of options. You can, for example, include various supplementary materials (such as introductions and appendices), change the default variable presentation "template", and include links to other HTML pages. It's a good idea to at least scan the SDA Manual sections that cover the XCODEBK program so you'll have some idea of the wide array of options available to you. A basic batch commands file can be very simple, however:

Example XCODEBK Batch Commands File

Assuming that the batch commands file and the variable list ('nvarlist') are in a subdirectory ('Docwork') of the study directory, here is an example of a command file:

type = HTML
study =..
varlist = nvarlist
savefile = ../Doc/nes

This batch file specifies that the output type should be HTML, the SDA study that will be documented is in the parent directory, the varlist is named “nvarlist” and is in the current working directory, and the output directory and “stem” name for the HTML files is “Doc/nes” (where 'Doc' is an already created subdirectory of the study directory). That’s all you need to create a basic, but quite useable, HTML codebook like those found on the SDA Web site. If you wanted to add some additional material – like an introduction and a couple of appendices, for example – you would just add a few more lines like the following:

intro = introfile
appendix = sample(**Sample Description)
appendix = weights(**Description of Weighting Procedures)
These specifications assume that the files 'introfile', 'sample', and 'weights' are in the current directory.

Note that by changing the "type = ..." specification, the XCODEBK program can produce codebooks formatted for printing. Before version 1.3 of SDA, these print codebooks were simple plain-text documents. However, SDA 1.3 introduced a method of creating codebooks in Microsoft Word format. The Word format allows the use of various fonts and styles, the introduction of graphical elements such as charts, and the dynamic updating of the table of contents. For details see the XCODEBK and WORDMACRO documents in the SDA Online Manual.

Run the XCODEBK program

After you have created the variable list and specified all the desired options in the batch commands file, you run the XCODEBK program. Assuming that the batch commands file is named 'mycommands.txt', just type the following command:

```bash
xcodebk -b mycommands.txt
```

(Note: in order to include frequencies and statistics in the codebook, you must create an SDA dataset before running the XCODEBK program. The XCODEBK program can generate a codebook based on only the DDL file, but these codebooks are generally far less useful. DDL-based codebooks should therefore only be created in special circumstances where you cannot use an SDA dataset as input or, for some reason, you do not want to include frequencies.)
Add Your Datasets to the Archive

Now that you've set up each of your datasets in SDA format and created an HTML codebook for it, you only have two simple tasks remaining to add each dataset to the Web data archive: add a new section in the "harcsda" file for it and add an HSDA startup link for it on your archive's "home page".

Add a section in the "harcsda" file for each dataset

The "harcsda" file already contains a listing for the sample dataset in the "[DATASETS]" section. You just need to add a listing for each new dataset you add to your archive. (Be sure to add a line with an asterisk in the first column between each dataset listing.) A minimal set of specifications for each dataset would typically include the following:

- DATASET = [Name of the study (one word)]
- DATALABEL = [Label of study to appear on options screens (one line)]
- CODEBOOK = [URL of "homepage" for HTML codebook]
- SDADATA = [PATH of SDA dataset directory]

If you're adding a "Job Satisfaction Study" to your archive your listing might look as follows:

```
DATASET   = jobsat
DATALABEL = Job Satisfaction Study
CODEBOOK  = http://myserver.mycollege.edu/data/jobsat/doc/jobs.htm
SDADATA   = /myserver/sdaweb/data/jobsat
```

For more information on these specifications (and others) be sure to read the HARC section in the SDA Manual.

Add an HSDA startup link for each dataset

You usually want to create an HSDA startup link for each dataset in your archive. For example, suppose the homepage for your archive starts with one link:

```
SDA Test Dataset
```

If you look at the HTML source for the link, you see something like this:

```
<a href="cgi-bin/hsda?harcsda+sdatest"> SDA Test Dataset </a>
```

Now you want to add another link for the "Job Satisfaction Study" you've added to the "harcsda" file:

```
Job Satisfaction Study
```

The link you add to the page would just substitute "jobsat" for "sdatest" and "Job Satisfaction Study" for "SDA Test Dataset":

```
<a href="cgi-bin/hsda?harcsda+jobsat"> Job Satisfaction Study </a>
```

Note that the "jobsat" name you use here must match the "DATASET" name you specified in the HARC file.

Now click on the link to your own dataset. You should see the initial options screen. Browse the HTML codebook for your dataset. Do some online analysis.

Congratulations, you've become an SDA Web data archive developer!
The Archiver: A Web-based Way to Archive Datasets

The preceding section has discussed the command line mode for adding datasets to your SDA archive. This method works well in many situations. However, there are situations where another, more interactive, approach is appropriate. The Archiver is a Web-based application that allows authorized users to add datasets to an SDA archive via a browser. The user just uploads a raw ASCII data file and a DDL (metadata) file via HTML forms. Options for codebook creation and archive configuration are also specified from the browser. The Archiver is designed to be very user-friendly; the technical knowledge required to archive datasets is greatly reduced in comparison to the command line mode.

Once the Archiver is installed, it is very simple to use (and it has extensive on-line help). Therefore we will not describe how to use the Archiver here. However, installing the configuring the Archiver does require some explanation. The following pages cover those tasks.

Deploy the Archiver "WAR" file

The Archiver application runs as a servlet and is distributed as a standard servlet "WAR" file. To run the Archiver on your server you must have installed a servlet container (such as Tomcat). It is quite possible that your organization has already installed a servlet container to run other Web applications (including SDA's chart-generation servlet.) If not, you can find more information on installing a servlet container in "Appendix B: Using the Tomcat Servlet Container". In the description that follows we will assume that you're using Tomcat as your servlet container and that you've configured Tomcat so that you can use its Web-based "Manager" application. With the Tomcat Manager you can deploy and undeploy, start and stop various servlet applications from your browser.

We'll assume that you have a copy of the Archiver "WAR" file ("sda_archiver.war") on your local PC or workstation. Using your browser, login to the Tomcat Manager application on your server. Now use the Tomcat Manager to deploy the "sda_archiver.war" file on your local disk to the server. (If you are updating a previously installed version of the Archiver, be sure to "undeploy" the current version from your server first; this will remove it.) Once the file is deployed, the compressed WAR file will be unzipped on the server so that its internal directory structure is visible.

Now use the Manager to temporarily stop the sda_archiver application on the server. You must do a little more configuration before the Archiver will function properly.

Create an Archiver Configuration File

The Archiver application must be set up with a configuration file on the server. A sample configuration file called "appconfig.txt" comes with the SDA distribution in the "archiver" directory. First, decide where you want to place this configuration file on your server. It can be in any location where it can be read by the Archiver application; it does not need to be in a document tree or virtual directory. Copy the "appconfig.txt" file to the desired location on your server and edit its contents to match the directory and URL paths on your server. (Note that a number of the keywords and specifications that go in this configuration file are the same as the items in a HARC file.) Next, you must specify the location of the "appconfig.txt" file in the Archiver's "web.xml" file (in the WEB-INF directory). Open the web.xml file with a plain-text editor (such as Notepad on Windows or vi on Linux/Solaris). Look for the <context-param> element that has the <param-name> "appconfig":


<context-param>
  <description>
    Runtime configuration properties file for application
  </description>
  <param-name>appconfig</param-name>
  <param-value>/mycomputer/mydirectory/appconfig.txt</param-value>
</context-param>

Now, inside the <param-value> element, edit the "/mycomputer/mydirectory" part of the path to match the actual location of the "appconfig.txt" file on your server.

For your reference, the following page lists the sample Archiver configuration file that comes with the SDA distribution. The file contains numerous comments (the lines starting with '#'): 
# Example configuration file for the Archiver

# WARNING: On Windows, the directory paths in this file must be specified
# using either a single forward slash (recommended) or double backslashes.
# A single backslash will cause an error. Examples:
# Correct: c:/mycomputer/sdaweb/archiverdir
# Correct: c:\mycomputer\sdaweb\archiverdir
# Error: c:\mycomputer\sdaweb\archiverdir

# Location of the parent directory where SDA studies will be archived. This
directory must be in a "document tree" or "virtual directory" on your server.
The "ARCHIVERDIR" specifies the directory path. The "ARCHIVERURL" specifies
the URL for the directory. These specifications are REQUIRED.
ARCHIVERDIR = /mycomputer/sdaweb/archiverdir
ARCHIVERURL = http://myserver.mycollege.edu/sdaweb/archiverdir

# The directory containing the SDA programs. (Same as HARC file specification.)
# This specification is REQUIRED.
SDAPATH = /mycomputer/sdaweb/sda

# HELPDOCS directory for SDA. (Same as HARC file specification.) This
# specification is REQUIRED.
HELPDOCS = http://myserver.mycollege.edu/sdaweb/helpfiles

# The URL of the CGI directory on your server that contains the SDA CGI
# programs (HSDA, HSDA2, etc.). This specification is REQUIRED.
CGI = http://myserver.mycollege.edu/cgi-bin

# Location of the log file for the archived studies. (Same as HARC
# file specification.) This specification is OPTIONAL (but recommended).
LOGFILE = /mycomputer/sdaweb/logs/sdalog

# Chartgen specifications. (Same as HARC file specification.)
# These specifications are OPTIONAL (but required for charts).
CHARTGENURL = http://myserver.mycollege.edu/chartgen
CHARTSTMPDIR = /mycomputer/sdaweb/tmpdir

# Search servlet URL. (Same as HARC file specification.)
# This specification is OPTIONAL (but required for searching).
SEARCHURL = http://myserver.mycollege.edu/sdasearch

# Subsetting specifications. (Same as HARC file specification.)
# These specifications are OPTIONAL (but required for subsetting).
SUBTMPDIR = /mycomputer/sdaweb/tmpdir
SUBTMPURL = http://mycomputer.mycollege.edu/sdaweb/tmpdir

# Specifications of maximum file upload size in megabytes. MAXUPLOAD specifies
# the maximum size of an uploaded file. MAXUNZIPPED specifies the maximum
# size of a compressed (zipped) file once it is unzipped on the server.
# These specifications are OPTIONAL.
MAXUPLOAD = 50
MAXUNZIPPED = 200

# Specification of file that contains locations of individual user
# configuration files. This specification is currently commented-out.
# See the Archive Developer's Guide for a discussion of this specification.
# USERCONF = /mycomputer/mydirectory/userconf.txt
**Archiver Security: Specifying Users/Passwords**

The Archiver application should now be configured so that it is password-protected. The file that contains the user-password pairs is in the WEB-INF/users.txt file. Here's an example "users.txt" file with three users:

```
dianne = emu, ROLE_ARCHIVIST
scott = wombat, ROLE_ARCHIVIST
henry = camel, ROLE_ARCHIVIST
```

The user names are on the left of each "=" sign. The password for each user is immediately on the right side of the "=". The role of the user ("ROLE_ARCHIVIST") follows the password (with an intervening comma). User names and passwords can be anything you find appropriate. You can define just one user or many. However, the "ROLE_ARCHIVIST" specification must be used as the role for all users you define. Note: the users and passwords specified in the users.txt file that comes with the Archiver distribution are just examples. You should replace them with your own users and passwords when you install the Archiver application on your server. Also, remember that each time you modify the WEB-INF/users.txt file after the Archiver has been installed you'll need to stop and restart the application before the changes will take effect.

You can use encrypted passwords in the WEB-INF/users.txt file for better security. See the comments in the included users.txt file for more information.

The Archiver's authentication mechanism is based on the popular open-source Spring Security project. See "Appendix C: Authentication and Spring Security" for more information.

**Specifying Additional Archiver Configuration Files for Individual Users**

By default, all users share the specifications found in the "appconfig.txt" configuration file that is specified in web.xml. In many situations this will be sufficient. However, you can provide user-specific configuration files that will override any or all specifications in the default "appconfig.txt" file. For example, suppose you want each user to store their datasets in a separate area on the server. That is, you want the "ARCHIVERDIR" and "ARCHIVERURL" specifications to be different for each user. Let's see how you would add additional configuration files to customize the Archiver for different users.

First, add a "USERCONF" specification to your "appconfig.txt" file. (This specification is contained in the sample "appconfig.txt" file but is commented out.) The file specified by "USERCONF" just contains a list of "username = file" specifications. For example:

```
# Location of individual configuration files for various users
dianne = /mycomputer/mydirectory/dianne.txt
scott = /mycomputer/mydirectory/datagroup.txt
henry = /mycomputer/mydirectory/datagroup.txt
```

The syntax of these user configuration files is the same as the default "appconfig" file. However, only the specifications that should override the default specifications need to be included in each user file. For example, the complete contents of the "dianne.txt" file might be:

```
ARCHIVERDIR = /mycomputer/sdaweb/diannedir
ARCHIVERURL = http://myserver.mycollege.edu/sdaweb/diannedir
```

This would mean that the archive contents for the user "dianne" would be written to a subdirectory called "diannedir" (instead of the default "archiverdir"). All other specifications would be taken from the default "appconfig.txt" file as
before. Note also that it is not necessary to create a unique file for each user. The users "scott" and "henry" both use the "datagroup.txt" configuration file.

**Restart the Archiver and Test It**

Once you've configured the Archiver following the above instructions you're ready to restart it and test it. Using the Tomcat Manager application, just find the "sda_archiver" in the list of deployed applications, then click on the "Start" link in the "Commands" column. To access the application, just click on the "sda_archiver" link in the "Path" column. You should now see a login screen. Use one of the user/password combinations you specified in the "users.txt" file to login. Now use the Archiver to add a new dataset to your archive. Extensive instructions are available from the Archiver's online help.
The New User Interface and the "Classic" Interface

The biggest change in the SDA 3.x series was the new user interface. However, the older, "classic" interface is still available (or its use is enforced) in various situations:

- A user can manually switch to the classic user interface by clicking on the "Use Classic Interface" link at the top of the new interface.

- The archivist can insert an "INTERFACE = CLASSIC" line in the [GENERAL] section of the HARC file. This will cause all the studies listed in that HARC file to start up using the classic interface.

- If the user's browser has JavaScript disabled, SDA will start up using the classic interface.

- If the "tree_items.js" file is missing from the directory containing the current study's HTML codebook files, SDA will automatically start up using the classic interface. (The tree menu of variables in the new interface depends on this "tree_items.js" file.) This ability to detect the absence of the "tree_items.js" file is, in turn, dependent on the URLCHECK specification in the HARC file not being turned off. By default, URLCHECK is on, but it can be disabled by inserting a "URLCHECK=NO" line in the [GENERAL] section of the HARC file.

Since the URLCHECK specification is linked to the ability of SDA to detect the presence or absence of the "tree_items.js" file (and other files accessed as URLs) we'll discuss it here. Some understanding of how URL-checking is done, and when it would be reasonable to turn it off, is useful for managing your archive. First, the URLCHECK specification controls whether checking is done in two circumstances:

- Checking whether the "tree_items.js" file exists for the current study when SDA starts up.

- In debug mode, the URLs specified in the HARC file (and some default URLs) are checked unless URLCHECK is set to "NO".

URL-checking is very handy while you're setting up or modifying your archive. Debug mode is much more useful when URLs are checked. Also, if you're upgrading from an earlier version of SDA and haven't yet re-created all your HTML codebooks, then it's very handy to have SDA just automatically switch to the classic interface for those studies that don't yet have a "tree_items.js" file. On the other hand, once your archive is up and running without problems, and all your HTML codebooks have been re-created using XCODEBK 3.x, then there's no necessity to keep URLCHECK turned on.

Finally, there are some very unusual circumstances where URL-checking might create problems and you will need to disable it. URL-checking uses a mechanism -- the XMLHttpRequest object that is the basis for the well-known AJAX Web programming model -- that cannot work if requests are made that violate its "same origin" policy. This means that the URLs that are checked must be in the same domain as that specified in the HSDA startup link. For example, if debug mode is invoked by specifying the domain www.mycollege.edu, then the URLs checked must also be from the www.mycollege.edu domain. The notion of "same domain" here is very strict. Not only must the domain names match, but any difference in protocol (http vs. https, for example), or port (80 vs. 8080, for example) will cause a problem. It is unusual for an SDA archive to be configured in a way that would violate this "same origin" policy. (Using different ports is probably the most likely scenario.) But if yours does, then just turn off the URLCHECK specification in your HARC file. You'll just need to be extra-vigilant that the URLs specified in your HARC file are correct and that you've created (or re-created) all your HTML codebooks with XCODEBK 3.x. (Note: there isn't a problem with running the chart-generation servlet from port 8080 – the Tomcat default port. URL-checking does not include the chart-generation servlet URL. However, it is recommended that you use a connector between Tomcat and your Web server so that you don't need to specify a separate port number. See the section in Appendix B below on "Connecting Tomcat to Apache or IIS" for more information.)
Search Configuration

Searching was the major new feature of SDA 3.2. Within a single study, variable-level searching is supported. In cross-study searching both variable-level and study-level searching is supported. These search capabilities are implemented by two servlets: the "searchmanager" and "sdasearch". The "searchmanager" is a web application that allows the archivist to initially configure the archive's search environment and create search indexes for studies. (Note that the searchmanager is only needed to set up the search environment; to save memory and other resources the servlet can be stopped until further configuration is required.) The "sdasearch" servlet runs continuously and handles all user searches. The sections below will discuss the deployment, configuration and usage of these two search-related servlets.

SDA's search capability is based on the popular open-source Apache Lucene project. More information on Lucene can be found at its Web site:

http://lucene.apache.org

Deploy the SEARCHMANAGER and SDASEARCH "WAR" files

The "searchmanager" and "sdasearch" servlets are distributed as standard servlet "WAR" files. To run these servlets on your server you must have installed a servlet container (such as Tomcat). It is quite possible that your organization has already installed a servlet container to run other Web applications. If not, you can find more information on installing a servlet container in "Appendix B: Using the Tomcat Servlet Container". In the description that follows we will assume that you're using Tomcat as your servlet container and that you've configured Tomcat so that you can use its Web-based "Manager" application. With the Tomcat Manager you can deploy and undeploy, start and stop various servlet applications from your browser.

Please note also that you should use a "connector" between Tomcat and your Web server (Apache or IIS) when configuring the "sdasearch" servlet. Some browsers, such as Firefox, have a very strict "same origin policy" that will not allow JavaScript to communicate between windows or frames loaded from URLs with different ports (i.e., 80 and 8080) – even if the host names are the same. This will cause the browser to not work properly when the buttons on the search results page are clicked. (Some versions of Internet Explorer, on the other hand, will work properly since it seems to have a slightly more relaxed "same origin policy" concerning port numbers.) There is more information on connecting Tomcat to Apache or IIS in "Appendix B: Using the Tomcat Servlet Container."

We'll assume that you have copies of the searchmanager.war and sdasearch.war files on your local PC or workstation. Using your browser, login to the Tomcat Manager application on your server. Now use the Tomcat Manager to deploy the searchmanager.war and sdasearch.war files from your local disk to the server. (If you are updating previously installed versions of the files, be sure to "undeploy" the current versions from your server first; this will remove them.) Once the files are deployed, the compressed WAR files will be unzipped on the server so that their internal directory structure is visible.

Now use the Tomcat Manager to temporarily stop the searchmanager and sdasearch applications on the server. You must do a little more configuration before the applications will function properly.

Create a Search Configuration File

Both the searchmanager and sdasearch servlets must be initialized with the same basic "searchconf.txt" configuration file on the server. A sample "searchconf.txt" file comes with the SDA distribution in the "search" directory. First, decide where you want to place this configuration file on your server. It can be in any location where it can be read by the searchmanager and sdasearch servlets; it does not need to be in a document tree or virtual directory. Copy the
"searchconf.txt" file to the desired location on your server and edit its contents to match the directory and URL paths on your server. (Note that a number of the keywords and specifications that go in this configuration file are the same as the items in a HARC file.) Here is the sample "searchconf.txt" configuration file that comes with the SDA distribution:

```
# searchconf.txt: Example configuration file for searchmanager and sdasearch

# WARNING: On Windows, the directory paths in this file must be specified
# using either a single forward slash (recommended) or double backslashes.
# A single backslash will cause an error. Examples:
# Correct: c:/mycomputer/sdaweb/sda
# Correct: c:\mycomputer\sdaweb\sda
# Error: c:\mycomputer\sdaweb\sda

# All of the specifications below are required.

# The directory containing the SDA programs. (Same as HARC file specification.)
sdapath = /mycomputer/sdaweb/sda

# The URL of the CGI directory on your server that contains the SDA CGI
# programs (HSDA, HSDA2, etc.).
# cgi = http://myserver.mycollege.edu/cgi-bin

cgi = http://myserver.mycollege.edu/cgi-bin

# Directory for temporary files. (Same as HARC file specification.)
subtmpdir = /mycomputer/sdaweb/tmpdir

# Directory for search configuration files that are written by searchmanager
# and read by sdasearch. The searchmanager will create two files in the
# directory: "datasets.txt" (which contains the datasets "registered" with the
# searchmanager) and "collections.txt" (which contains information about
# the collections that are defined with the searchmanager).
confdir = /mycomputer/searchconfdir

```

Next, you must specify the location of the "searchconf.txt" file in both the searchmanager and sdasearch "web.xml" files. Open each WEB-INF/web.xml file with a plain-text editor (such as Notepad on Windows or vi on Linux/Solaris). Look for the `<context-param>` element that has the `<param-name>"searchconf"`:

```
<context-param>
  <description>
    Runtime configuration properties file for application
  </description>
  <param-name>searchconf</param-name>
  <param-value>/mycomputer/mydirectory/searchconf.txt</param-value>
</context-param>
```

Now, inside the `<param-value>` element, edit the "/mycomputer/mydirectory" part of the path to match the actual location of the "searchconf.txt" file on your server.

**SEARCHMANAGER Security: Specifying Users/Passwords**

The searchmanager (unlike sdasearch) needs to be configured so it is password-protected. The user/password information is in the searchmanager's WEB-INF/users.txt file. Here's an example user/password specification:

searchman=mysecret,ROLE_ARCHIVIST
The user name is on the left of the "=" sign. The password is immediately on the right side of the "=". The role of the user ("ROLE_ARCHIVIST") follows the password (with an intervening comma). The user name and password for the searchmanager can be anything you find appropriate. However, "ROLE_ARCHIVIST" must be used as the role specification. Note: the user and password specified in the users.txt file that comes with the searchmanager distribution is just an example. You should replace them with your own user and password when you install the searchmanager application on your server. Also, remember that each time you modify the WEB-INF/users.txt file after the searchmanager has been installed you'll need to stop and restart the application before the changes will take effect.

You can use encrypted passwords in the WEB-INF/users.txt file for better security. See the comments in the included users.txt file for more information.

The searchmanager's authentication mechanism is based on the popular open-source Spring Security project. See "Appendix C: Authentication and Spring Security" for more information.

Use the SEARCHMANAGER to Configure the Search Environment

Now that the searchmanager application has been deployed and configured, it can be used to set up your archive's search environment. For single-study searching the searchmanager is used to "register" each dataset and create a search index for it. For cross-study searching the searchmanager is used to define "collections" (groups of datasets). The searchmanager comes with extensive documentation to help you use it. But here are a couple of things to keep in mind when using the searchmanager.

Permissions and the searchmanager: it is important to note that the searchmanager servlet can only do its work if it has adequate read/write/execute permissions. In particular, the searchmanager must be able to:

- **Read** the HARC file(s) that are used to register datasets.
- **Write** search indexes in the STUDYINF directory of SDA datasets. The searchmanager will create a "SEARCH" sub-directory under the STUDYINF directory, and then write the search index information within that directory.
- **Execute** the programs in the "sda" directory – including the utility programs "harcrdr" and "vartext".

Study-level information: the searchmanager will look for study-level information (for cross-study searching) in an SDA dataset's STUDYINF directory. The searchmanager will look for study-level information in the STUDYINF directory in the following order:

1. It will look for a DDI (Version 2) file called "ddi.xml". The file must have this exact name for the searchmanager to find it. If a "ddi.xml" file exists the searchmanager will read selected elements from it and store the information in a file called "study-properties.txt".

2. If the DDI file can't be found, the searchmanager will look for a file called "study-properties.txt". The file must have this exact name for the searchmanager to find it.

3. If neither a "ddi.xml" nor a "study-properties.txt" file can be found, then the searchmanager will use the study's title (found in the "studyinf" file) as the only available study-level information to add to the search index.

For you, the archivist, this means that if you have a DDI file for your study, you can simply copy it to the study's STUDYINF directory (and rename it "ddi.xml" if necessary). If you don't have a DDI file, then you can create a "study-properties.txt" file and copy it to the study's STUDYINF directory. Unlike DDI, the format of this file is very simple. (The format is documented in the searchmanager help file.) Finally, if you don't provide either file, the searchmanager will just use the study title as the only available study-level information to add to the search index.
**Use the SDASEARCH Servlet for Single-Study or Cross-Study Searching**

Once your archive's search environment is set up with the searchmanager, the sdasearch servlet handles either single-study or cross-study searches.

**Single-study searches:** you just need to add a "SEARCHURL" specification to the [GENERAL] section of a HARC file. For example:

```
SEARCHURL = http://myserver.mycollege.edu/sdasearch
```

The SEARCHURL just specifies the URL of the "sdasearch" servlet.

**Cross-study searches:** you just need to put a link to the sdasearch servlet on one (or more) or your site's pages. For example:

```
<a href="http://myserver.mycollege.edu/sdasearch"> Search All Collections </a>
```

In this default mode all the collections defined via the searchmanager will be displayed. Alternatively, the "clist" (collection list) syntax can be used to limit (and reorder) the collections displayed. For example, suppose you've specified the following collection IDs using the searchmanager: 'gss', 'anes', 'census', 'multi', 'proj'. If you want to create a link that will only display the 'anes' and 'census' collections (in that order), then you can create a link like the following:

```
<a href="http://myserver.mycollege.edu/sdasearch?clist=anes,census"> Search the ANES and Census Collections </a>
```

To specify a 'clist' add "?clist=" after the usual sdasearch URL, then specify the collection IDs (separated by commas but no spaces). The 'clist' syntax can be useful if you want to provide multiple cross-study searching links on your site that are customized for particular sections of your site that deal with different subsets of collections.

**A Note on the Archiver and Searching**

The Archiver (since SDA 3.2) has been enhanced so that it can enable within-study searching for a dataset. To turn on this feature the Archiver's configuration file ("appconfig.txt") must include a "SEARCHURL" keyword that specifies the URL of the "sdasearch" servlet. For example:

```
SEARCHURL = http://myserver.mycollege.edu/sdasearch
```

This SEARCHURL specification is copied from the "appconfig.txt" configuration file to the study's HARC file when the dataset is archived. (If you want to add search to a study that was created by the original version of the Archiver, you can just start up the new Archiver, enter the Study ID on the first tab, then go to the final "Archive" tab and click on "Finish Archiving Study".)

The Archiver must be able to set up within-study searching independently -- without the involvement of the Search Manager or the search configuration files it manages ("datasets.txt" and "collections.txt"). The Search Manager may not be running or -- in some scenarios -- the users of the Archiver may not be allowed access to the Search Manager application. Therefore, the Archiver application itself will create a search index for a study when it is archived. Also, in order to avoid the need to "register" Archiver-created studies with the Search Manager, the Archiver instead creates a special "SEARCHSTUDY" keyword in the [DATASETS] section of the study's HARC file. This SEARCHSTUDY specification contains the path to the SDA dataset (like SDADATA). It is also a flag to HSDA that it should render the SDA interface so that the search servlet is invoked with alternative parameters that allow searching even though the dataset is "unregistered".
This special handling of search configuration in the Archiver does not preclude using the Search Manager at a later time to "register" Archiver-created datasets and add them to collections. Archiver-created studies can be managed by the Search Manager in the same manner that any other SDA studies are managed. Just remember that each Archiver-created study will have its own HARC file called "harc.txt" in the study's main directory (named after the study ID).
Appendix A: Upgrading to SDA 3.5 from an Earlier Version

This appendix provides instructions on how to upgrade to SDA 3.5 from an earlier version of the software. (For information on what has changed between versions, consult the "Updates" section in the online SDA Manual.)

First, please note that each new version of SDA has been backward-compatible with previously-created SDA datasets. Updating your current SDA installation does not require you to update your SDA datasets. Other issues to consider depend on which version of SDA you currently have:

**If you are upgrading from a pre-3.x version of SDA:**

SDA 3.0 introduced a new user interface. To take advantage of this major enhancement you will eventually need to re-create your HTML codebooks using a 3.x version of XCODEBK. Why? The 3.x versions of XCODEBK not only create the usual HTML codebook files. These newer XCODEBK versions also create a special file – always called "tree_items.js" and located in the same directory as the HTML files – that provides the necessary information to create the variable tree menu in the lower left corner of the new user interface. The "tree_items.js" file, like the index of headings in the HTML codebook, is structured according to the headings that are found in the variable list when XCODEBK is run. The re-creation of the HTML codebooks in your archive is actually the biggest task in upgrading to the newest version of SDA from a pre-3.x version. See below for the other steps to take.

For most archives, there will be no need to make other changes to HARC file(s) or HSDA startup links. But there have been a few minor changes in these areas. First, it is no longer legal to use an HSDA startup link that does not include a dataset name (or 'debug') after the HARC file name. (This startup mode is incompatible with the new user interface.) Also, there have been a few changes to the HARC syntax. Some keywords that were previously optional are now required. In the [GENERAL] section, HELPDOCS is now required. In the [PROGRAMS] section, SDAPATH and SDAPROGS are now required. In the [DATASETS] section, DATALABEL, CODEBOOK, and SDADATA are now required.

**If you are upgrading from a 3.x version of SDA:**

Since you have (presumably) already re-created your HTML codebooks for the new user interface, the upgrade process is quite straightforward. Just update your SDA files following the instructions below.

**Steps to upgrade to SDA 3.5:**

- Download the appropriate distribution package for your platform (Linux, Windows or Solaris). Unzip or uncompress the distribution package but DON'T directly overwrite your current SDA files. Instead, place the new version in a currently unused area on your disk.

- Before updating your SDA files, make a backup copy of the files in your current "cgi-bin", "sda", and "helpfiles" directories (or their equivalents if you’ve changed their names). It's unlikely that anything will go wrong but, if it does, you can now easily restore your old setup.

- Now, with one exception, copy the contents of the "cgi-bin", "sda", and "helpfiles" directories in the new SDA version over your current files. The one exception is the "harcsda" file in the "cgi-bin" directory. If you have added your own datasets to that file (instead of creating a HARC file of your own with a different name), you don't want to overwrite your old "harcsda" file. The "harcsda" file distributed with the SDA software is intended to be a template for new users. If you are already running SDA, you won't need that file. **Warning:** be sure to copy all the files and subdirectories in the "helpfiles" directory to your archive setup. The "helpfiles" directory now contains a "jscript" subdirectory with several crucial files. (And the "jscript" subdirectory in turn contains an "icons" subdirectory.) Also, the "sda" directory contains recent utility programs introduced in version 3.2 and 3.3 -- "harcrdr", "vartext", and "subset". Be sure copy them to your archive's "sda" directory too.
• **If you're upgrading from a pre-3.0 version:** now that you've updated your SDA archive with the newer versions of the files in the "cgi-bin", "sda", and "helpfiles" directories, you should re-create the HTML codebook for each of the datasets in your archive. Once you've re-created a dataset's HTML codebook with the new XCODEBK program, you should see the new SDA interface when you click on the HSDA start-up link for that dataset. What happens if you don't re-create the HTML codebook for a dataset? You will see the older, "classic" SDA interface. When **HSDA** starts up it looks for the "tree_items.js" file in the directory that contains the HTML codebook files. If **HSDA** can't find the file, it automatically reverts to the older user interface since it can't create the variable "tree" for the new user interface.

• **Install or upgrade SDA’s servlet-based webapps:** once you've upgraded the rest of your SDA installation, you can install (or upgrade) the servlet-based SDA webapps. These webapps can be found in the "servlets" subdirectory of your SDA distribution and include chartgen, the search webapps, the archiver and quicktables. **Please note: for SDA 3.5 all of the servlet-based webapps have been updated with enhancements and/or bug fixes; we would advise that you update any older versions on your system.** To install the servlet-based webapps consult the appropriate documentation. For chartgen, consult the section "Adding Charts to Tables Output" in this guide. For search, follow the instructions given in the "Search Configuration" section in this guide. For the archiver, see the section "The Archiver: A Web-based Way to Archive Datasets" in this guide. For "Quick Tables", see the "readme.txt" file in the "quicktables" subdirectory of the distribution; it includes a link to online documentation and an example configuration file. **Please note that the SDA 3.5 servlet-based webapps require version 6 of the Java Development Kit (JDK) and version 6 of Tomcat.**

• Once you've installed the new SDA files (and re-created your HTML codebooks if necessary), you're done. If you encounter problems, see Appendix D: Troubleshooting – particularly the section on using debug mode.
Appendix B: Using the Tomcat Servlet Container

Version 1.4 of SDA introduced the capability of creating charts. Version 3.1 introduced the Archiver. Version 3.2 introduced searching. Version 3.4 introduced Quick Tables as a standard part of the SDA distribution for the first time. However, taking advantage of these capabilities requires having a Java servlet “container” running on your server. Some ACS member organizations are already running a servlet container to support other applications and are therefore already familiar with servlet technology. However, for organizations that are unfamiliar with obtaining and configuring a servlet container, this appendix provides some information about the Tomcat servlet container. Why Tomcat? Other servlet containers are available and, as long as they’re standards-compatible, they can be used with SDA instead of Tomcat. However, other servlet containers are often part of larger Web application server software suites and may have significant costs. Unless your organization has a compelling reason to run another servlet container, Tomcat is probably your best choice since it’s free, widely-used, high quality and strictly compliant with the servlet specification.

Tomcat History

Tomcat started as a project within Sun Microsystems, the inventor of Java, to provide a reference implementation for the Java servlets specification. Servlets are an alternative to CGI for providing dynamic Web pages and applications. Sun later donated the Tomcat project (and code) to the Apache Software Foundation – which also houses the Apache Web server project. Just as Apache is the most widely used Web server, Tomcat is the most widely used servlet container. Both are developed as free, open-source software.

Installing and Configuring Tomcat

Installing Tomcat (and a Java JDK/JRE)

Please note that SDA 3.5 requires Java version 6 and Tomcat version 6.

Tomcat requires that you first have a Java Development Kit (“JDK”) or Java Runtime Environment (“JRE”) installed on your machine. (Installing the JDK is probably the more common choice.) As of Sept. 2010 a Java 6 JDK could be downloaded at: http://www.oracle.com/technetwork/java/javase/downloads/index.html. Follow the directions available on the site for downloading and installing the JDK.

Once you have a JDK or JRE installed you can download and install Tomcat. Tomcat binaries are available at: http://tomcat.apache.org. Follow the download and installation directions available on the Tomcat site for your platform.

Configuring Tomcat’s Memory Usage

Prior to version 5 of Sun/Oracle's Java Virtual Machine (JVM) – part of the JDK/JRE -- Tomcat was only allocated a maximum 64MB of memory by default. Consequently, it was often necessary to increase the available memory by editing Tomcat’s configuration files. However, a version 5 or later Sun/Oracle JVM will configure itself according to the type of machine it’s installed on. If the machine has two or more physical processors and two or more GB of memory, then it will default to the “server” version of the JVM. The server JVM will automatically set a maximum heap size of ¼ of physical memory up to 1 GB. Therefore, it is much less likely today that you will need to configure Tomcat’s memory usage – and Sun/Oracle recommends that you do not manually set the configuration unless you are sure that you need more memory. (By the way, if you’re using Tomcat you can monitor Tomcat’s JVM memory usage by using the Tomcat Manager application. Look for the “JVM” heading on the “Server Status” page.)

If you are sure you need to manually increase Tomcat’s memory usage, you can do so by editing the “catalina.sh” file on Linux/Unix or the “catalina.bat” file on Windows. (These files are in the “bin” directory within the Tomcat installation.) Suppose you want to allocate a beginning minimum of 128MB and a maximum of 256MB of memory to Tomcat. On Linux/Unix, you’d add this line at the top of the “catalina.sh” file:
CATALINA_OPTS = -Xms128m -Xmx256m

On Windows, you’d add this line to the “catalina.bat” file:

set CATALINA_OPTS = -Xms128m -Xmx256m

You can see that the “Xms” argument sets the minimum allocated memory and the “Xmx” argument sets the maximum memory.

**Deploying an Application with Tomcat**

“Deploying” an application just means installing a servlet web application in a servlet container. Servlet web applications are typically packaged as “WAR” (Web ARchive) files that conform to a standardized directory/file layout and are then compressed in a zip format with a “.war” suffix. All of the SDA servlet-based webapps conform to this "WAR" packaging standard. There are a number of ways of deploying a “WAR” file with Tomcat. However, Tomcat includes a Web-based "Manager" application that makes deploying WAR files – and performing other management tasks – quite easy. So let's look briefly at enabling the Tomcat Manager now.

**Enabling the Tomcat Manager**

To enable the Tomcat Manager you need to configure its password protection. By default, Tomcat uses the user/password information stored in the "tomcat-users.xml" file in the "conf" directory of your Tomcat installation. (You can configure other methods of storing user/password information – such as a database -- but we will just cover the default method here to get you started. See the Tomcat documentation for more information.) In the `<tomcat-users>` element you must have a defined role of "manager" and at least one user who has that role. A minimal `<tomcat-users>` element that satisfies these requirements might look like this:

```xml
<tomcat-users>
  <role rolename="manager" />
  <user username="manager" password="mysecret" roles="manager" />
</tomcat-users>
```

The username can be whatever you like – it just must include "manager" in the list of "roles". And, of course, you will want to provide your own password. Once you've set up the password protection (and restarted Tomcat) you can go to the Tomcat home page, click on the "Tomcat Manager" link and login to the Manager. From there you can access the various management functions via HTML forms in your browser.

Before leaving this topic, we will briefly describe a couple of easily-implemented techniques for improving the security of the Manager webapp: 1) encrypting the password and 2) limiting access to only specified IP addresses.

**Encrypting the password:**

Instead of storing the manager password in clear text in Tomcat's conf/tomcat-users.xml file, we can easily improve security by encrypting the password. Here we'll use the SHA (Secure Hash Algorithm) method – although Tomcat supports a number of other encryption types. First, in Tomcat's conf/server.xml file, set the "digest" attribute of the `<Realm>` element to "sha":

```xml
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
       resourceName="userDatabase" digest="sha" />
```

Now create an encrypted version of your password by using Tomcat's "digest" script in the "bin" directory. On Linux/Unix use the script named "digest.sh"; on Windows, use "digest.bat". The algorithm to use (SHA) and the string to digest (a password in this case) must be specified as parameters. On Linux/Unix for example, the command to get a SHA-encrypted version of "mysecret" is:
The output is the clear text password, followed by a colon and the encrypted password:

```
mysecret:e9fe51f94eadabf54dbf2fbbd57188b9abee436e
```

Now add the encrypted password to the `tomcat-users.xml` file. The digested password simply goes in place of the clear text password:

```
<tomcat-users>
  <role rolename="manager" />
  <user username="manager" password="e9fe51f94eadabf54dbf2fbbd57188b9abee436e" roles="manager" />
</tomcat-users>
```

**Limiting access to only specified IP addresses:**

You can configure Tomcat so that it will allow access to a webapp only from specified IP addresses. You can easily use this mechanism to secure the manager webapp. In Tomcat's `webapps/manager/META-INF/context.xml` file, add a `<Valve>` element within the existing `<Context>` element:

```
<Context . . .>

  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\0.0\1,123\45.678\90" />

</Context>
```

The `<Valve>` element must use the "className" attribute shown above. The "allow" attribute specifies a comma-delimited list of IP addresses that will be allowed to access the webapp. In the example above the first IP address is just "localhost" – the generic name and number (127.0.0.1) that allows a machine to reference itself. It is followed by a comma and the (fictitious) IP address of another machine (123.45.678.90) that will also be allowed to access the manager. In your manager's `context.xml` file, list the IP addresses of all the machines that you want to have access to the application. If you try to access the manager from an unlisted machine, you'll just get a "403" error saying that access to the specified resource has been forbidden.

Why are the periods in the IP addresses escaped with a backslash? The IP addresses allowed can be specified as regular expression patterns that follow the rules of the Jakarta regex library. The periods are therefore escaped with backslashes so they won't be considered wildcard characters.

**Connecting Tomcat to Apache or IIS**

It's possible to use Tomcat and a Web server together without a connector. Tomcat can just be accessed from its default port 8080 and the Web server (Apache or IIS) from the standard port 80. For example, if you're using the chartgen servlet to generate charts, you could just specify something like the following in your HARC file:

```
CHARTGENURL = http://myserver.mycollege.edu:8080/chartgen
```

However, there can be difficulties with this approach. A few users are behind very restrictive firewalls that block "non-standard" ports like 8080; these users will see a "broken link" image – rather than a chart – in their analysis output. Also, some browsers, such as Firefox, enforce a very restrictive "same origin" security policy that will not allow JavaScript to communicate between windows or frames loaded from URLs with different ports (i.e., 80 and 8080) –
even if the host names are the same. This will cause the browser to not work properly when the buttons on the search results page are clicked. Therefore it is often necessary to use a connector between Tomcat and your Web server. In this configuration all requests first go to the Web server (on port 80). The Web server then checks the request's URL pattern to see if the request should be sent on to Tomcat for processing. To the user, all requests look like they're handled by the "fronting" Web server on port 80 – although Tomcat is doing its usual work in the background via a connector. When Tomcat is used with a connector, the servlet URLs are specified without the appended port 8080:

CHARTGENURL = http://myserver.mycollege.edu/chartgen

The connection between Tomcat and a fronting Web server (Apache or IIS) is typically done via an AJP-based connector. AJP stands for Apache Jserv Protocol. (Jserv was the name of an early Apache servlet container project that was dropped in favor of Tomcat.) The current production version of AJP is 1.3 – the version that comes with Tomcat 6. On the Tomcat side, AJP is implemented as a connector that is configured via Tomcat's server.xml file. The Tomcat AJP connector specification looks something like this:

<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

Once you've ensured that a valid AJP connector element is defined in the server.xml file, you're usually ready to go on the Tomcat side. Most of the connector work involves configuring the Web server side of the connection. On the Web server side, the AJP protocol is typically implemented as a shared object module for Apache (mod_proxy_ajp, for example) and as a DLL for Windows' IIS (isapi_redirect.dll).

Unfortunately, connecting Tomcat to a Web server has sometimes been a difficult task. And the available documentation has often been incomplete or out of date. For example, recommendations of the "next generation" JK2 connector can still be found in various online documents and books – but, in fact, the JK2 connector is now deprecated and is no longer being developed. Recently the situation has improved. The newer versions of Apache – in the 2.2.x series – have made connector configuration much easier. And a couple of recent Tomcat books provide some much-needed, up-to-date documentation:

- **Professional Apache Tomcat 6**, Chopra, Li, Genender (Wrox 2007). Covers configuration of both the Apache-Tomcat connector and the IIS-Tomcat connector. If you're looking for help on connecting IIS and Tomcat this is the book to consult.
- **Tomcat: The Definitive Guide (for Tomcat 6)**, Jason Brittain (O'Reilly 2008, 2nd ed.). Covers the Apache-Tomcat connector options in some detail. It also contains a lot of information about the performance of various configurations. **However, it does not cover IIS to Tomcat connector configuration at all.**

If you consult these books, be sure to get the most recent editions (covering Tomcat 6). The earlier versions are now badly out of date. Both books go into a great deal more detail on the connector issue than we can here. But we will give you a very quick overview of the procedure – mainly based on the material in these books. We'll first cover connecting Apache to Tomcat, then IIS to Tomcat.

### Apache to Tomcat Connector

Until fairly recently, connecting Apache to Tomcat usually involved configuring a JK connector – which was implemented as an Apache module called (appropriately enough) mod_jk. However, using mod_jk can be difficult. The mod_jk module is distributed by Tomcat (not Apache) but must be compiled against your exact version of Apache or it won't run. Various binary versions of mod_jk are available from the Tomcat Web site but it is often difficult to determine which Apache version they are compatible with. Therefore you must often build mod_jk yourself from source. This situation has sometimes been described as "mod_jk hell" and is best avoided if possible.

Luckily, there is another choice for fronting Tomcat with Apache: mod_proxy (and its helper modules such as mod_proxy_ajp and mod_proxy_http). If you are using the latest Apache – the 2.2.x series -- then mod_proxy is clearly the preferred method since it is both simpler to configure and, according to **Tomcat: The Definitive Guide**, 43
significantly faster. The mod_proxy module is built into Apache httpd 2.2.x. It often comes ready to use with Apache releases (especially on Linux).

In sum, according to Tomcat: The Definitive Guide: "…our benchmarks show that mod_jk is the slowest connector we used to connect Apache httpd with Tomcat. And … mod_jk is also the most difficult to get working. We cannot suggest that you download, build, install, configure, and use mod_jk as opposed to simply adding settings to your httpd's configuration file [to use mod_proxy]…" (pg. 190). If you are forced to use an older version of Apache then you may need to use mod_jk -- which we will not cover here. Consult one of the books above -- and the Tomcat documentation -- for more information on using mod_jk.

The mod_proxy modules that are built into Apache httpd can, in fact, be used with either the AJP or HTTP protocols. Either protocol can work well -- although AJP is somewhat faster and probably more commonly used. We'll briefly cover both below.

**Using mod_proxy with the AJP protocol:**

**Configuring the Apache side:**

Load the appropriate modules. Presuming the mod_proxy modules are already available, you just need to add (or uncomment) the following lines in the appropriate place in your Apache httpd.conf file.

```bash
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
```

Next, specify the URL pattern requests that Apache should send on to Tomcat for processing. Let's suppose that any request for /sdasearch should be sent to Tomcat -- which is running on the host "myserver.mycollege.edu". You would add the following to your httpd.conf file, below the LoadModule lines:

```bash
ProxyPass /sdasearch ajp://myserver.mycollege.edu:8009/sdasearch
ProxyPassReverse /sdasearch ajp://myserver.mycollege.edu:8009/sdasearch
```

Note the protocol (ajp) and the port (8009).

Similar pairs of lines should be added for each servlet that you want to use with the connector: chartgen, sda_archiver, etc.

**Configuring the Tomcat side:**

If you're using the AJP protocol, you need to enable a <Connector> element for it in the server.xml file. (The default server.xml file has an AJP Connector element already created, but it may be commented out.) The Connector element for the AJP protocol should look something like this:

```xml
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

(Note that the port must match the port specified in the ProxyPass and ProxyPassReverse directives in the Apache httpd.conf file.)

Now restart Apache and try using the servlets you've configured using the default Apache port 80 (instead of Tomcat's 8080).
Using mod_proxy with the HTTP protocol:

Configuring the Apache side:

Load the appropriate modules. (Note that on the second line below, the "mod_proxy_http.so" module is loaded instead of "mod_proxy_ajp.so").

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

Specify the URL pattern requests that Apache should send on to Tomcat for processing.

```
ProxyPass /sdasearch http://myserver.mycollege.edu:8080/sdasearch
ProxyPassReverse /sdasearch http://myserver.mycollege.edu:8080/sdasearch
```

Note that only the protocol (http vs. ajp) and port numbers (8080 vs. 8009) are different than the AJP protocol configuration above.

Similar pairs of lines should be added for each servlet that you want to use with the connector: chartgen, sda_archiver, etc.

Warning: be sure that you specify URL patterns using either the AJP or HTTP protocol -- **but not both**!

Configuring the Tomcat side:

If you're using the HTTP protocol, you probably don't need to do anything since there should already be a Connector for the HTTP protocol in Tomcat's server.xml file.

Now restart Apache and try using the servlets you've configured using the default Apache port 80 (instead of Tomcat's 8080).

Troubleshooting: Linux and mod_proxy connector

We ran into the following problem and, since the solution is somewhat obscure, thought we'd pass on our experience.

**Problem:** you're on a Linux system and the mod_proxy connector doesn't work (using either the HTTP or AJP protocol) and you find errors like these in the Apache error log file:

```
[Tue March 25 17:19:54 2008] [error] (13)Permission denied: proxy HTTP: attempt to connect to 128.32.119.43:8080 (myserver.mycollege.edu) failed

[ Tue March 25 17:19:54 2008] [error] ap_proxy_connect_backend disabling worker for (myserver.mycollege.edu)
```

**Solution:** this is likely a problem with the SELinux default security settings. Try using the following "setsebool" command to allow mod_proxy:

```
setsebool -P httpd_can_network_connect true
```

(The -P flag has to be added to make the change permanent.)
IIS to Tomcat Connector

As we noted above, the IIS-Tomcat connection uses the AJP protocol. On the Tomcat side the usual AJP Connector handles things. On the IIS side an ISAPI plug-in component is used. (ISAPI stands for Internet Server Application Programming Interface.) The Tomcat Web site provides an ISAPI plug-in called "isapi_redirect.dll" (with a version number embedded in the download file's name). This ISAPI plug-in functions as both a filter – which watches request URLs for patterns that should be routed to Tomcat -- and an IIS extension – which forwards request parameters to Tomcat and gets the response back.

The configuration "recipe" below is mainly based on the information in chapter 12 of Professional Apache Tomcat 6, entitled "Tomcat and IIS." For more detailed instructions (including helpful screen shots) you should consult the book -- which contains the clearest and most in-depth account of this procedure we have found. The Tomcat Web site also contains helpful documentation on connecting IIS and Tomcat; it is not nearly as user-friendly but it may be adequate for some. Note that the Tomcat documentation varies somewhat from Professional Apache Tomcat 6 – mainly in where the ISAPI Plug-in is stored and the names/locations of various other components. You can follow either source's suggestions – just make sure you follow one naming/location convention or another so your setup is internally consistent.

Check that Tomcat's server.xml configuration file has an AJP Connector.

The ISAPI Plug-in will use an AJP connector. The element specification in Tomcat's server.xml will look something like this:

```xml
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
```

Create directories for storing and configuring the ISAPI Plug-in

Professional Apache Tomcat 6 suggests storing the Plug-in within the IIS directories rather than the Tomcat directories (as the Tomcat documentation suggests). That makes sense since it is IIS – rather than Tomcat – that uses the Plug-in. So we'll follow the book's advice on this point.

Under the C:\Inetpub directory create an "ISAPI" directory. Then create three sub-directories of ISAPI: "bin", "conf" and "logs. You should now have these new directories:

- C:\Inetpub\ISAPI\bin
- C:\Inetpub\ISAPI\conf
- C:\Inetpub\ISAPI\logs

Download the ISAPI Plug-in

Download the latest production-quality ISAPI Plug-in and save it in the "C:\Inetpub\ISAPI\bin" directory. At the time of this writing (April 2008), the ISAPI Plug-in binary for 32-bit versions of Windows was available at:


Under this directory the most recent stable version was jk-1.2.26. In the appropriate JK Connector directory you will find the most recent version of the ISAPI plug-in. Currently it is: isapi_redirect-1.2.26.dll. Note that the name of the plug-in includes the version number (1.2.26). After you have saved the plug-in to your "C:\Inetpub\ISAPI\bin"directory, rename the plug-in to the simpler "isapi_redirect.dll". That will make your configuration easier and allow you to easily update the plug-in version later if desired.
Create a "workers.properties" file

Create a "workers.properties" file in the "C:\inetpub\ISAPI\conf" directory. This file specifies the Tomcat "workers" (instances) that your connector configuration will utilize. Presuming you only have one Tomcat running, your "workers.properties" file would look something like this:

```plaintext
worker.list = tomcat1
worker.tomcat1.type = ajp13
worker.tomcat1.host = localhost
worker.tomcat1.port = 8009
```

The "worker.list" just lists the name(s) of the tomcat instance(s) you're running. The name(s) can be anything you find useful. In this case we just use "tomcat1" to make things simple. The following lines specify the type (ajp13), host (localhost) and port (8009) for the "tomcat1" worker. These settings should work fine as long as your Tomcat is running on the same machine as your IIS and you're using the default port of 8009 for the AJP Connector.

Create a "uriworkermap.properties" file

Create a "uriworkermap.properties" file in the "C:\inetpub\ISAPI\conf" directory. This file specifies the URL patterns that will cause IIS to route requests to Tomcat. Here's the contents of a "uriworkermap.properties" file that handles the "sdasearch" and "chartgen" webapps:

```plaintext
/sdasearch/* = tomcat1
/chartgen/* = tomcat1
```

The specification on the left side of the '=' gives the URL matching pattern. The specification on the right side of the '=' gives the name of the tomcat worker (instance) that should handle it. The URL-matching patterns are explained in the online Tomcat documentation (and the Professional Apache Tomcat 6 book). The pattern shown above, with a '|' ("pipe"), then a "/*" is a very handy specification. The "/sdasearch/*" specification will match "/sdasearch", "/sdasearch/*" and also anything longer such as "/sdasearch/search". This is often just what you want.

Update the Windows Registry

The ISAPI Plug-in needs to have settings created for it in the Windows Registry. Start up the Registry editor by typing "regedit" in a commands window. Before you change the Registry you should make a backup copy by using the editor's File->Export option. (You can restore the backup later, if necessary, by using the File->Import option.) Now navigate to the HKEY_LOCAL_MACHINE->SOFTWARE key. Your Tomcat installation should already have created an "Apache Software Foundation" key. If not, create it now. Then create the "Jakarta Isapi Redirector" and "1.0" sub-keys. **Be sure that your key names match the following names exactly:**

```
HKEY_LOCAL_MACHINE -> SOFTWARE -> Apache Software Foundation -> Jakarta Isapi Redirector -> 1.0
```

Now, under the "1.0" key, create the following mandatory parameters (as String values):

```plaintext
extension_uri = /tomcat/isapi_redirect.dll
worker_file = C:\inetpub\ISAPI\conf\workers.properties
worker_mount_file = C:\inetpub\ISAPI\conf\uriworkermap.properties
```

In addition, it is helpful to create the following settings so a log file is created. The log_level should be reduced to "warn" or "error" (from "debug") once the connector is working.

```plaintext
log_file = C:\inetpub\ISAPI\logs\iis_redirect.log
log_level = debug
```
Switch to IIS 5 Isolation Mode (IIS 6 only)

If you're on a Windows Server 2003 (and therefore using IIS 6) you need to switch to IIS 5 Isolation Mode. Follow these steps:

1) Right-click and select Properties on the Web Sites node in the IIS Manager.
2) Select the Service tab in the Properties page.
3) Check the check box for "Run WWW service in IIS 5.0 isolation mode".
4) Click "Apply", then restart IIS.

Create a Virtual Directory Under IIS

In the IIS Manager right-click the Default Web Site node and select New->Virtual Directory. This will start the "Virtual Directory Creation Wizard". Click "Next" which will take you to a form that asks for the Virtual Directory alias. Type in "tomcat" then click "Next" again. Select the directory where the ISAPI Plug-in is located: "C:\Inetpub\ISAPI\bin". On the next screen – "Access Permissions" – make sure the following checkboxes are selected: "Read", "Run scripts (such as ASP)"), "Execute (such as ISAPI applications or CGI)", "Browse". Once the Wizard is completed, the Virtual Directory "tomcat" should be displayed as a sub-branch.

Add the ISAPI Plug-in as an IIS Filter

In the IIS Manager right-click on the Default Web site and select Properties. Click the ISAPI Filters tab and click the Add button. For "Filter Name" use "isapi_redirect". For "Executable" use "C:\Inetpub\ISAPI\bin\isapi_redirect.dll". Click "Apply" and then restart IIS.

Authorize ISAPI Plug-in as a Web Application Extension (IIS 6 Only)

Follow these steps:

1) Click on Web Service Extensions from the IIS Manager
2) Right-click and select "Add a new Web Service Extension"
3) Enter a name for the extension – "isapi_redirect" – and click "Add"
4) Select "C:\Inetpub\ISAPI\bin\isapi_redirect.dll"
5) Check the Set extension status to Allowed check box

Check the connector

Now restart IIS and try using the servlets you've configured using the default IIS port 80 (instead of Tomcat's 8080).
Appendix C: Authentication and Spring Security

The SDA Archiver and Searchmanager web applications need to be password-protected. The authentication mechanism supplied in the SDA package is based on the open-source Spring Security project. (Spring Security was originally called Acegi, but changed its name when version 2 was released. The Archiver and Searchmanager webapps were upgraded to Spring Security 2 from Acegi in the SDA 3.3 release.) Spring Security is part of the widely-used Spring Framework, a Java/J2EE application platform. More information on Spring Security can be found at its Web site:

http://static.springframework.org/spring-security/site

Spring Security was chosen because of its portability and flexibility. First, it will work not only with Tomcat, but with any standard servlet container. Other security systems -- including Tomcat's -- are typically vendor-specific. Second, Spring Security offers a broad choice of authentication options. Although the user names and passwords are defined in the WEB-INF/users.txt file in the SDA distribution, Spring Security can be configured (via external XML files) so that the user/password information comes from another source – such as a database or LDAP server. Although the simple users.txt file is probably sufficient for most archive's needs, you can consult the Spring Security Web site for information on other options for storing user names and passwords.

Finally, you can replace Spring Security with another authentication mechanism if you wish. Just comment out – or remove – the section in the WEB-INF/web.xml file that configures Spring Security. Then configure the authentication method you prefer.
Appendix D: Troubleshooting

DEBUG MODE

The first thing to remember when troubleshooting your SDA Archive is: **debug mode is your friend.** This is especially true in SDA 3.x since debug mode's diagnostic capabilities have been significantly enhanced in this version. Debug mode was mentioned earlier in this Guide, but we'll cover it again here since it's so important.

First, to enable debug mode you must put a "DEBUG = yes" line at the start of the "[GENERAL]" section of your HARC file. (By default debug mode is disabled.) To invoke debug mode you simply substitute the word "debug" in place of the dataset name in an HSDA startup link. For example, a startup link might look like this:

http://www.mycollege.edu/sdaweb/cgi-bin/hdsa?harcsda+mystudy

The dataset name in this link is 'mystudy'. To invoke debug mode, just substitute 'debug' for 'mystudy':

http://www.mycollege.edu/sdaweb/cgi-bin/hdsa?harcsda+debug

A debug mode link will return a page of diagnostic information about the HARC file – in this case the file "harcsda" – and the specifications it contains. Any syntax errors will be reported. Also, debug mode will check for file and directory access problems – for example, an inability to access the directory where the SDA analysis programs reside or an inability to write temporary files to the appropriate directory. Beginning with SDA 3.0, debug mode also checks that all the necessary URLs can be accessed (as long as URLCHECK has not been set to "NO" in the HARC file). This includes not only the codebook HTML pages, for example, but also the JavaScript library files, CSS files, etc. that the new user interface require. Finally, debug mode will dump out a listing of all the information contained in the HARC file – both the explicit specifications and the implicit (default) specifications.

IIS and Windows Server

Over the years, the default configuration for IIS/Windows Server has more and more become "lock-down mode" – where everything is denied that is not explicitly permitted. Also, Microsoft is obviously not primarily interested in supporting traditional CGI for dynamic Web applications; their focus is on their own Microsoft-specific solutions. The result is that the most recent version of IIS – IIS 7 – has become so difficult to configure for a sophisticated CGI application that we cannot recommend its use with SDA. We recommend that SDA users on the Windows platform either use IIS 6 or Apache for Windows. Apache for Windows is quite straightforward to configure (much like the versions for Linux/Unix) and supports CGI very well. IIS 6 is more difficult to configure, but still manageable. In the section below we walk through the configuration of IIS 6/Windows Server 2003 for SDA. And, after that, we make a few more observations about IIS 7.

IIS 6 and Windows Server 2003

Installing SDA on IIS6/Windows Server 2003 requires a number of non-obvious configuration steps. To help out, we'll briefly list the typical steps needed to set up a basic SDA Web archive on this platform:

- Copy the "sdaweb" folder (and all its contents) into the default IIS home directory "C:\Inetpub\wwwroot". The SDA home page (with links to the SDA Test Dataset and Debug mode) should now be visible.
- Change execute permissions for the "cgi-bin" folder. From the IIS Manager:
  - Right-click on the "cgi-bin" folder and choose "Properties"
  - Go to the "Directory" tab and change "Execute permissions" to "Scripts and Executables"
- Create a C:\tmp folder (for various SDA-produced temporary files) if it doesn't already exist
• Edit the SDA distribution's "harcsda" HARC file to change paths and URL's as needed. (See the comments in the "harcsda" file for more information on this.)

• Enable "Anonymous Access". From the IIS Manager:
  o Right-click on "Default Web Site"
  o Click on "Properties"
  o Click on "Directory Security" tab
  o Click on the "Edit …" button in the "Authentication and Access Control" panel. The name shown in the "user name" input box should be "IUSR_<computer_name>". If not, select that user.
  o Click on the check-box for "Enable anonymous access"

• Enable "Web Services Extensions". From the IIS Manager:
  o Click on the "Web Services Extensions" folder
  o Choose "All Unknown CGI Extensions"
  o Click on "Allow"

• At this point the "Debug mode" link should work. Be sure to inspect the "Debug mode" output and correct any errors in the "harcsda" configuration file that are reported.

• There is another configuration change needed – although several features of your SDA archive should now work. You can browse the HTML codebook, download the existing dataset and documentation, and the analysis option forms work correctly. However, you will never see the actual analysis output; you just get a blank screen. Also, if you've enabled the "customized subsetting" option, the subset ASCII data file will be created correctly, but the accompanying codebook and data definition files (SPSS, SAS, STATA, DDI) will be empty. The problem is that the SDA CGI programs are unable to "call" or execute the SDA programs in the "sda" directory (such as TABLES, XCODEBK, DDLTOX, etc.) Calling an external program in Windows depends on read/execute access to the Windows command processor "cmd.exe" (usually in \windows\system32). Microsoft Knowledge Base Article 311481 entitled "PRB: IIS 6.0: CGI Code That Calls External Applications May Fail" discusses this issue. As of March 2008 this article was available at: http://support.microsoft.com/default.aspx?scid=kb;en-us;311481. The upshot is that you need to grant read/execute permission to the "cmd.exe" program for the IUSR_<computer_name> account. To do so:
  o In Windows Explorer, go to the \windows\system32 folder. Right click on cmd.exe and choose properties. Click on the security tab. (If you don't see a security tab, then open "Folder Options" in Control Panel. On the View tab, under Advanced settings, clear "Use simple file sharing [Recommended]"). Click on the "Internet Guest Account" in the top pane. Click on "Read & Execute" in the allow column, (Read will automatically be selected, and there will just be a selection for "write" in the deny column). Click on OK. Your basic SDA archive should now work properly.

• To enable other capabilities that require the CGI programs to write files – subsetting, chart generation, etc. – you will need to allow the "Internet Guest Account" to write files in the necessary directories. To do this: in Windows Explorer right-click on a directory name, click on "Properties", click on the "Security" tab, click on "Internet Guest Account", then click on the "Allow" checkbox in the "Write" row.

• Also, to enable the SDA features that are servlet-based, you will need to deploy the appropriate WAR file(s) – chartgen.war, sdasearch.war, etc. – to a servlet container such as Tomcat. For help on this, including connecting your Web server to Tomcat, see "Appendix B: Using the Tomcat Servlet Container".

Some Notes on IIS 7

As stated above, we cannot recommend using IIS 7 with SDA. However, if there are SDA users who, for one reason or another, are curious about trying that route anyway, we will just make a couple of brief observations. First, IIS 7 and its Manager application have been radically re-designed. If you were comfortable with IIS 6 you can nevertheless expect to have a whole new learning experience with IIS 7.

Next, as far as we can determine, you must individually configure each CGI program – hsda.exe, hsda2.exe, hsda3.exe, subset1.exe, subset2.exe, subset3.exe – in the IIS Manager so that it is allowed to execute. However, even after you have successfully enabled the individual CGI programs to execute, you will nevertheless encounter the following error message when initiating a typical SDA startup link:

   “Hyper-link format is incorrect: no HARC filename was specified.”
This is puzzling, since you can see you have, in fact, specified a HARC file and dataset name. For example, your startup link might look something like this:

http://www.mycollege.edu/sdaweb/cgi-bin/hsda.exe?harcsda+sdatest

What is going on? It turns out that IIS 7 apparently does not allow ‘+’ (plus) characters in a URL’s query string – the “harcsda+sdatest” part of the URL above. Therefore, the “hsda.exe” CGI program is never passed the query string from IIS 7 and never sees the HARC file name. Once it was determined why this failure was occurring, we tweaked the code in SDA 3.5 to allow the more verbose, but admittedly more customary, syntax for query strings:

key1=value1&key2=value2&key3=value3 ……

So in SDA 3.5 the keys “harc”, “dataset”, and “outstudy” are now recognized. The example startup link above can instead be expressed as:

http://www.mycollege.edu/sdaweb/cgi-bin/hsda.exe?harc=harcsda&dataset=sdatest

Now the query string “harc=harcsda&dataset=sdatest” will be accepted by IIS 7 and passed on to the hsda.exe CGI program. Once you’ve managed to make the SDA startup link work, you may find that much of SDA works as expected. However, in our own efforts to configure IIS 7 we have not been able to get RECODE and COMPUTE to work correctly within the IIS environment. This may be due to various reasons: perhaps there is a security setting somewhere that still needs to be tweaked – or perhaps there is a bug in IIS that’s causing a problem. Whatever the reason, at this point, after spending a significant amount of time reading the IIS 7 documentation and Googling for clues, we cannot recommend that anyone else take this path. (And from reading about other people’s experiences while researching these CGI issues in IIS 7, we know we are not alone.) It simply shouldn’t be this hard to configure a CGI webapp. There are other, better, choices out there for CGI support in a Web server than IIS 7.

**VARCASE SPECIFICATION**

The "VARCASE" specification for a dataset in the HARC file can be very useful. If all the variables in a dataset are upper case, for example, it’s very helpful for users if a "VARCASE = upper" line is specified. Users can then enter variable names in either upper or lower case – or even mixed case – and still get the results they expect. However, it is very important that the VARCASE specification – "upper" or "lower" – is set correctly. If you specify the wrong case then you will run into mysterious problems that debug mode doesn't detect. So if VARCASE is specified for a dataset, make sure it's set correctly. And remember, if the variables in a dataset aren't all the same case (with the exception of the CASEID variable), VARCASE cannot be specified at all for that dataset.

**CHARTS ON LINUX OR SOLARIS**

If the "chartgen" servlet fails on your Linux or Solaris system with Xserver related error messages such as:

java.lang.InternalError: Can't connect to X11 window server using ':0.0' as the value of the DISPLAY variable

then you have encountered the Xserver "headless" problem. For more information about this, see the discussion and links in the JFreeChart FAQ #10 at http://www.jfree.org/jfreechart/faq.html. The upshot is that you may need to start up Tomcat with the flag:

-Djava.awt.headless=true

For more background information on this issue see the Sun article on Java and "headless" mode:

FOR MORE INFORMATION …

The troubleshooting tips above will often solve a problem with your archive. However, for more detailed information about SDA's various programs, be sure to consult the online SDA Manual (available on the "Documentation" page at the SDA Web site). If all else fails, you can always email the SDA staff at sda@berkeley.edu.