Ministry of Community Services and
Ministry of Tourism, Sport and the Arts

Ministry Software Configuration Management Policies and Procedure for Source Material Management

Date: March 27, 2007
Prepared By: David Ell
Project: CAWS Standards Documentation
Harvest Package: RFC_000452
Harvest Version: 14

Table of Contents

Revision History

Date	Harvest Version	Section	Description	Author
04-APR-2004	0	All	Initial Draft	David Ell
13-JUL-2004	1	All	Second Draft integrating DWS group feedback	David Ell
13-JUL-2004	2	TP and TOC	Update Title Page and TOC	David Ell
22-JUL-2004	3	All	Third Draft integrating DWS group feedback	David Ell
22-JUL-2004	4	All	Fourth Draft integrating DWS group feedback	David Ell
22-JUL-2004	5	2.1.2.1	Add language about testing	David Ell
28-JUL-2004	6	All	Revised with Ted Dixon's changes	David Ell
30-JUL-2004	7	All	Fifth Draft integrating DWS group feedback	David Ell
04-AUG-2004	8	All	Sixth (and Final) Draft	David Ell
17-AUG-2004	9	2.3.1	Fixed typographical error	David Ell
04-OCT-2004	10	2.2.2.2	Amended policy	David Ell
18-OCT-2004	11	Approver	Updated Approver and Title	David Ell
18-OCT-2004	12	Footers	Changed modified date stamp – {SAVEDATE} Added Document Properties	K Warnes
27-MAR-2007	14	changed	Ministry of Community Services and Ministry of Tourism, Sport and the Arts	Kwarnes

Document Approval

Signature	Date
Dave Kent	Manager, Development and Web Services
Print Name	Title

1.0 Overview

1.1 Outline

This document is being compiled to outline the policies and procedures to be used by developers who are using the Ministry's Harvest Software Configuration Management (SCM) repository to manage project source materials. For the purposes of this document, source material is defined as any/all source code and/or dependency files which are used during compilation and/or execution of an application.

Within the policies stated below there are sections, by language/compiler, which define how to manage source materials with the Harvest SCM repository.

1.2 Background

These policies have arisen out of the project to define a software configuration management methodology for the Ministry. These policies were created to inform developers of what the Ministry considers to be appropriate source code management techniques, using the Harvest SCM repository.

1.3 In Scope

The following topics are in scope for this document:

- Appropriate level of repository interaction
- Source code management
- Application development policy for source material management
- Source materials that are being managed using SCM

1.4 Out of Scope

The following topics are not in scope for this document:

- The Ministry's Software Configuration Management methodology
- Database component management
- Coding standards
- Architecture standards
- Source materials that are **NOT** being managed using SCM

1.5 Assumptions

This document assumes a reader has already received training on the Ministry SCM methodology as well as on the use of the Harvest change management tool. It is also assumed that the reader has the Harvest Workbench installed on their desktop. If you have not received Harvest training, or do not have the Harvest Workbench installed, contact the Ministry Business Analyst for the project.

2.0 Policies and Procedures

2.1 All source materials are to be stored in the Ministry Harvest Repository.

2.1.1 Reason For This Policy

This policy has been created to ensure that the Ministry, as the owner of project deliverables, possesses all project deliverables during the entire Software Development Life Cycle (SDLC). This will allow for more efficient project transitions, as well as allowing the Ministry to perform Quality Assurance (QA) checks of applications during the development process.

2.1.2 Procedure

Use the methods described in the sections below to Check In/Out code to/from the Harvest Repository.

2.1.2.1 All File Types

This section describes procedures which apply to all source material file types that developers will be checking in to the Harvest repository.

- All non-standard file extensions are to be reported to the Harvest Administrator and designated as either binary or text format. This information is required by the Harvest tool in order to allow the most efficient repository compression.
- All referenced dependency files (.dll, .txt, .csv, .ocx, etc…) are to be checked into the Harvest Repository.
- All test scripts and/or data conversion scripts are to be checked into the Harvest Repository.
- Files are to exist in only one directory within the repository. **There must NOT be multiple copies of a file, stored in different directories.**
- As a best practice, files should be stored using the structure below. Exceptions can be made where adoption of this Repository structure is not practical. If required, sub-directories can be made under the Code folder to mimic the implementation environment.

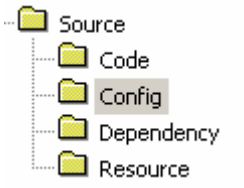

Figure 1: The Source directory and its default sub-structure

Code: Contains all source code files (.frm, .frx, .java, .vb, etc…) which are used in the compilation of a system.
Config: Contains all configuration files which are referenced by a system. (i.e. .ini or .xml files)
Dependency: Contains any external binaries (i.e. .dll, .ocx, etc…) which are referenced by a system.
Resource: Contains any data files (.txt, .csv, .mdb, .dbf) which are used as a resource by a system. See Section 2.2 for additional procedures regarding .dbf and .mdb files.

2.1.2.2 Visual Studio 6, Visual Studio .NET 2002 and 2003

This section describes procedures regarding the checking in and out of Visual Studio files.

- Code files are to be Checked In/Out through the IDE only. For information on Check In/Out of non-code files, see Section 2.1.2.3 for instructions on using the Windows Explorer-integrated components of the Harvest Workbench.
- Once the Harvest client has been installed on a machine which has Visual Studio installed, it will automatically integrate into the IDE.

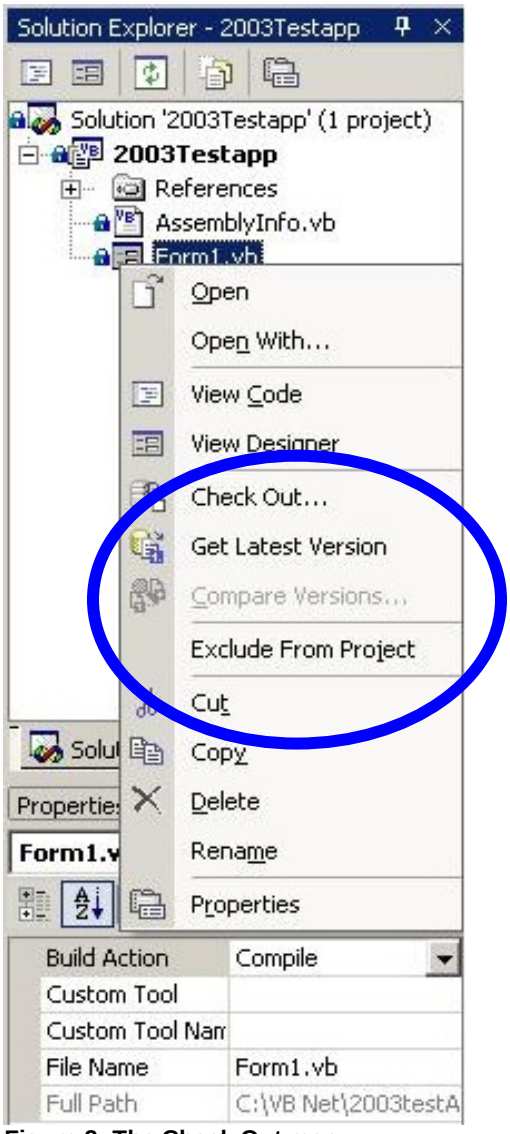

Figure 2: The Check Out menu

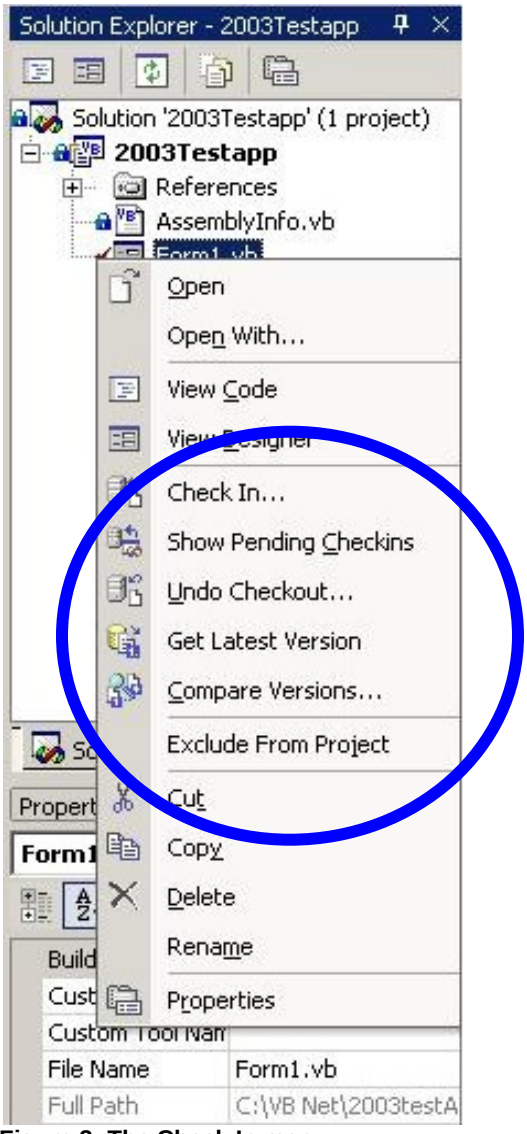

Figure 3: The Check In menu

Warning: The Harvest client will over-write any active SourceSafe connections established on a system. If you require both Harvest and SourceSafe access, see Appendix B for further information.

2.1.2.3 All non-Visual Studio Development

For all non-Visual Studio files, the following procedures should be followed:

- Once the Harvest client has been installed, it will integrate with Windows Explorer. This will allow you to Check In/Out your files as well as choosing which RFC/Package to work against.

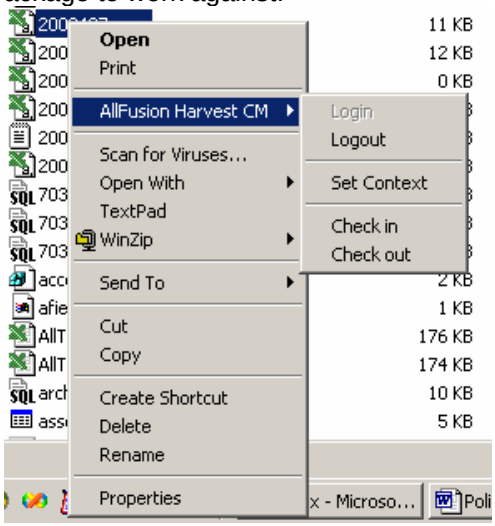

Figure 4: Windows Explorer integration

- If you are checking in a file which is part of the compiled code for an application, de-select the "Delete file on check-in" option (refer to Figure 5 below). If this option is not de-selected, the application may no longer compile due to missing files. The file will then need to be refreshed from the Workbench, using a Check Out for Browse.

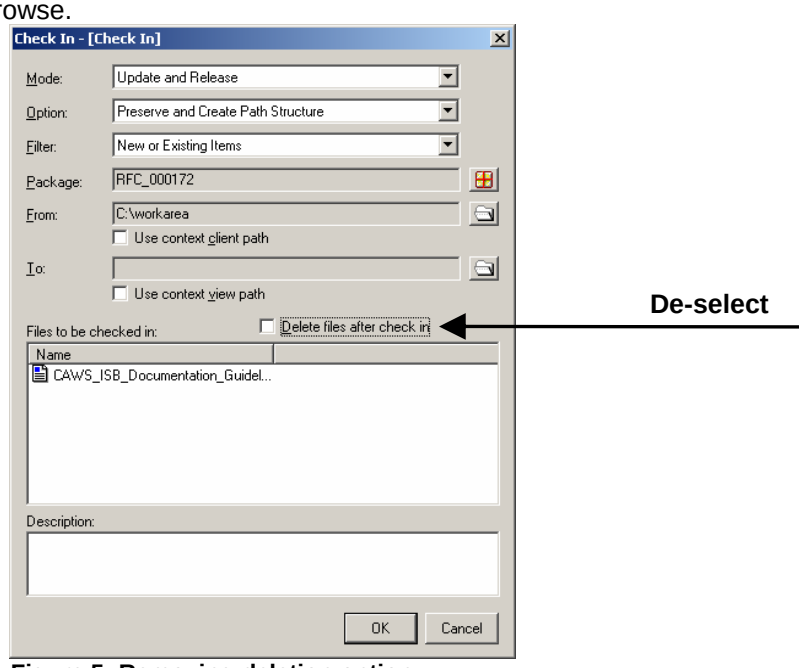

Figure 5: Removing deletion option

Warning 1: The Harvest client will over-write any active SourceSafe connections established on a system. If you require both Harvest and SourceSafe access, see Appendix B for further information.

Warning 2: The menus illustrated in Figure 4 will only appear if you have performed a custom installation of the Harvest Workbench (as per the Ministry's installation instructions). If this menu does not appear, you will need to uninstall and reinstall the Workbench.

2.2 All source material changes are to be tracked through the Harvest Repository.

2.2.1 Reason For This Policy

This policy allows for a very fine level of change management to be achieved. This will give programmers (as well as internal QA staff) a very explicit description of what code has been modified in order to complete a Request For Change (RFC). This also allows for packages to be removed from a release with relative ease, as well as simplifying roll-back procedures.

2.2.2 Procedure

All Check In/Out processes must be conducted in reference to a RFC or Deficiency package within Harvest. If no RFC or Deficiency packages have been created, consult with the project BA to have them created.

Use the guidelines described in the sections below to ensure that your changes are tracked to an appropriate level.

2.2.2.1 Visual Studio 6, Visual Studio .NET 2002 and 2003, Java, ASP, JSP, VBScript, non-binary FoxPro files, and JavaScript

These procedures will apply to all text-based file types:

- At the beginning of the development process for a RFC, Check Out the appropriate files, modify them as required for a single RFC, and then check the files back into the Harvest repository.
- Each file should be checked in once development and unit testing (of a given RFC) is **complete**. This will ensure that only functional modules exist within the Repository.

2.2.2.2 Microsoft Access, Lotus Approach, and binary FoxPro files

These procedures will apply to all binary file types:

- All MS Access databases are to be **compacted prior** to being checked in.
- Files of a binary format are to be checked in only at one point in the Software Development Lifecycle. The file is to be checked-in at the start of the delivery process within the Integrated System Testing state.
  NOTE: Additional Check-In/Out cycles may be required; however, this best-practice will help to keep storage requirements to a minimum. This is due to the fact that the large binary files created by these applications are not modularized, and thus require repeated storage of unchanged material.

- All MS Access and Lotus Approach data tables are to have their data purged, prior to being checked in to the Harvest repository. This will allow for documentation of the structure, without the storage burden of the data itself. Lookup (or reference) tables should be stored with their records intact. Some systems may have specific test data which may be useful to keep and re-use. In this case a full system check in is acceptable.

2.2.2.3 Other Tools and File Types

Before beginning with development, consult with the Harvest Administrator to discover what frequency of Check In/Out is appropriate.

2.3 All source materials are to be stored in a recoverable environment.

2.3.1 Reason For This Policy

While the Ministry is providing a backed-up repository for completed changes, it is still the responsibility of the developer to back-up code which is currently under development. This will reduce time lost due to catastrophic system failure.

2.3.2 Procedure

All source materials, which are currently checked out of the Harvest repository are to be backed-up nightly. Source materials should NOT be stored on the C: drive of a developer's computer. For Windows Forms .NET development, since the source files under development must be on the developer's C: drive, it is the responsibility of the developer to ensure that files are transferred to a backed-up location (such as a network drive) on a nightly basis.

2.4 All coding is to be conducted in a non-concurrent manner.

2.4.1 Reason For This Policy

This policy has been created to ensure that developers do not get into situations where they are attempting to merge the work of multiple developers. Merging multiple version branches does not lead to a high-quality final product and is not an efficient development methodology. Branch merging can also put the code base at risk (i.e. previously resolved issues may re-emerge in subsequent releases).

2.4.2 Procedure

Only a single developer is allowed to work on a given file at a time. The Harvest repository will only allow a single check-out of a file to assist with the enforcement of this policy. If two developers wish to work on the same module, only one of them is to have the file checked-out for update at any given time. It is recommended that such files be further modularized to assist in eliminating conflicts.

2.5 All Applications should display both Release number and Build number.

2.5.1 Reason For This Policy

Applications should display their Release and Build numbers to provide an obvious determination between different releases of an application. See Appendix C Glossary for further definitions of Release and Build numbers.

2.5.2 Procedure

On the interface of all applications it should be possible to display both the Release and Build numbers of an application. This is commonly done through the Help...About box or Splash Screen of a Windows Forms-based application (i.e. MS Access, VB6, VB.NET, etc…), or in a discrete location on the interface of a Web-based application.

An application's implementation is uniquely identified by the combination of the Release and Build numbers (i.e. R01.00.00, Build 5). The Release number should be a value that is stored in the application's database. If there is no database, or the database is proprietary, then the Build number should be stored in a shared constant in the application code. The Build number is either compiler-provided, or hard-coded using a shared constant (eg. gconstBUILD_NUMBER), to allow for distinction between application builds. See Appendix A for examples.

Appendix A Release and Build Numbers

The following are examples of placement and styles for Release and Build numbers.

Windows Forms-based Applications:

Figure 6: Example of Release and Build Numbers on a Splash Screen.

Web-based Applications (BC Government Portal Look and Feel):

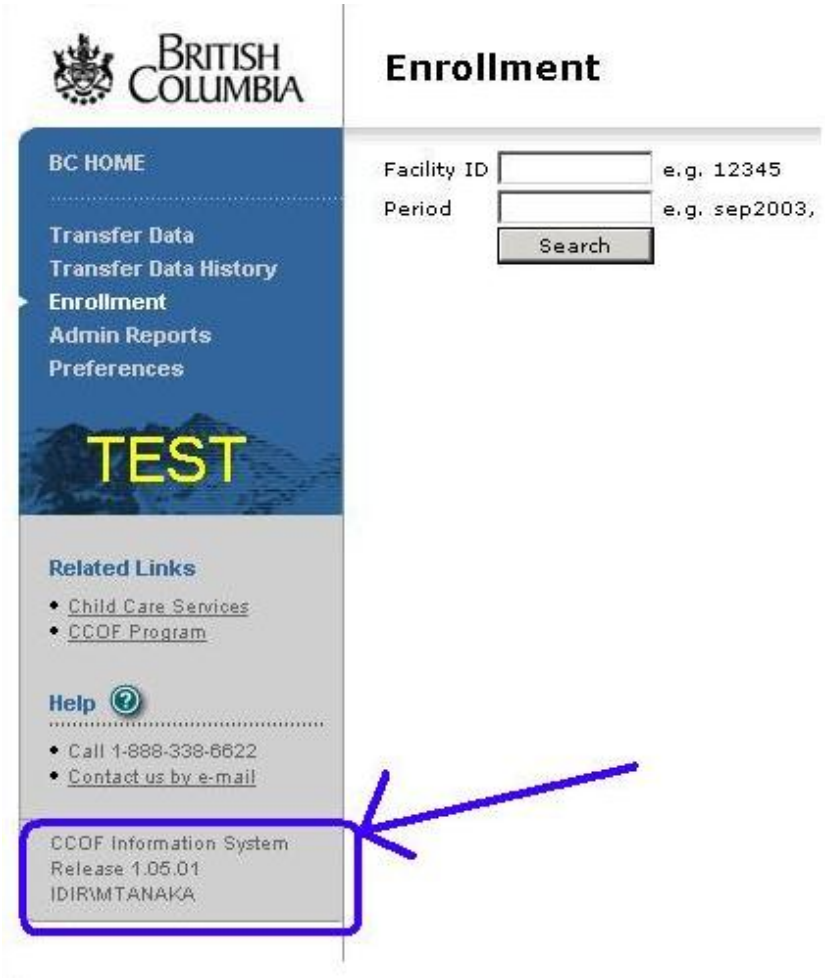

Figure 7: Example of Release number on a BC Government Portal-style page.

Note: This example is missing the Build number. The Build number should be placed immediately below the Release number

Web-based Applications (Non-Portal Look and Feel):

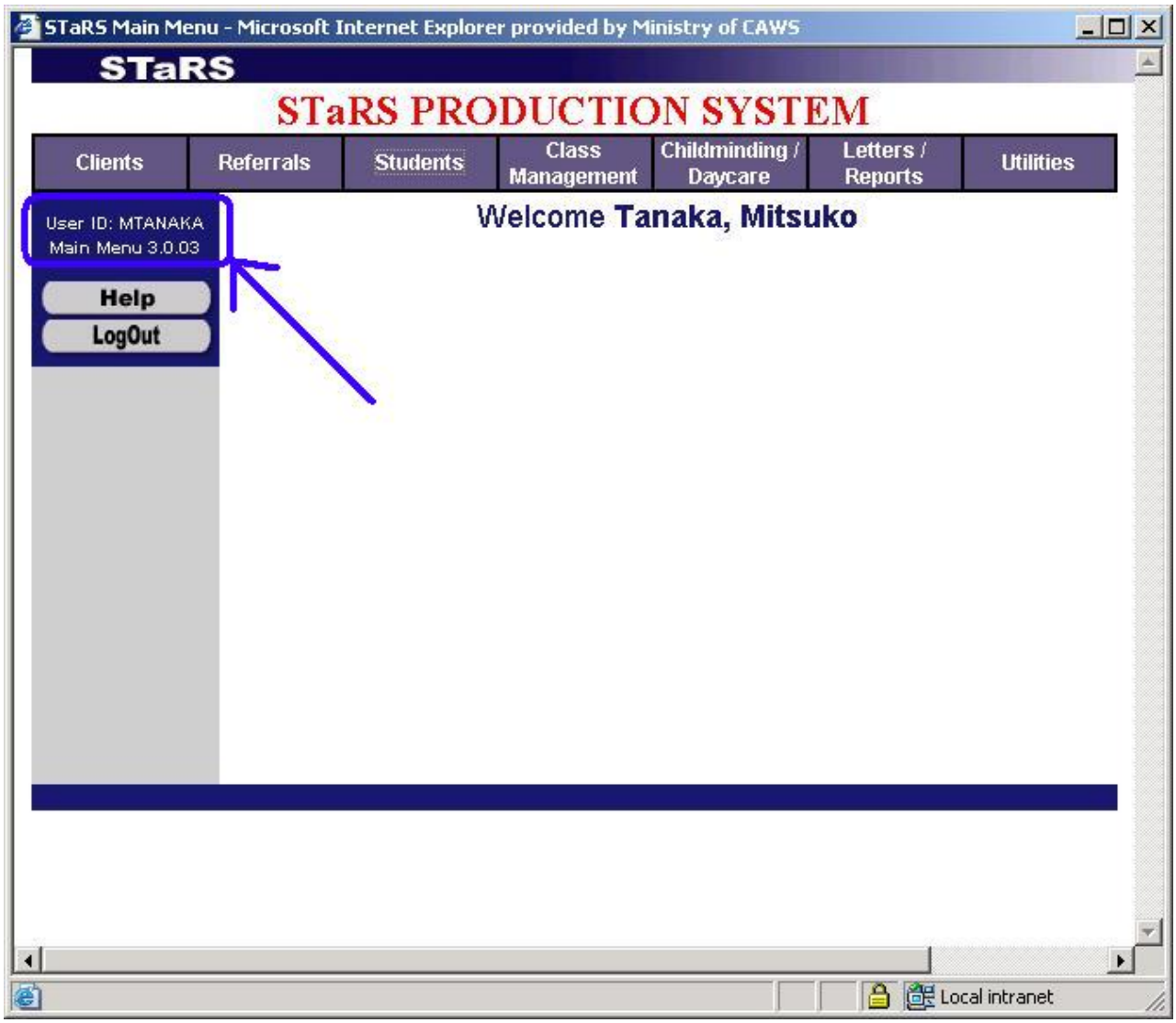

Figure 8: Example of Release number on a custom web-application page.

Note: This example is missing the Build number. The Build number should be placed immediately below the Release number

Appendix B The SCC Switch Utility

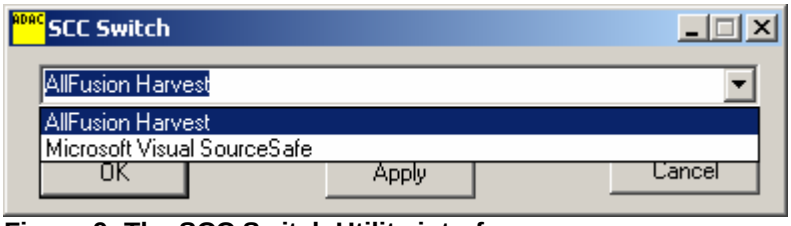

Figure 9: The SCC Switch Utility interface

The SCC Switch utility will allow you to switch source code providers on a computer. When you run the utility it will default to the opposite provider from what you are running (i.e. if you are running SourceSafe it will default to Harvest, and vice-versa). This utility is available from the Harvest Administrator; however, we will only provide best-effort support

Appendix C Glossary

Acronym / Term	Description
ASP	Active Server Pages
BA	Business Analyst, as defined in the Ministry SCM methodology.
Build	A Build of an application uniquely identifies a single successful compilation and linking of all source code. For a web-based application it is assigned by the programmer to uniquely identify implementations of source code.
Harvest	The Ministry standard tool for SCM
IDE	Integrated Development Environment (ex. Visual Studio .NET 2003)
IST	Integrated System Testing
JSP	Java Server Pages
MS	Microsoft
QA	Quality Assurance
Release	A unique identifier for a single iteration through the SDLC. (Eg. R01.00.00) The Release number is used to bind all project deliverables under a single identifier.
RFC	Request For Change (a logical unit of work)
SCC	Source Code Control, the Microsoft standard API for integrating with a source material management system.
SCM	Software Configuration Management
SDLC	Software Development Life Cycle
Source Material	Any source code and/or dependency files that are used during compilation and execution of an application.
Version	A unique identifier for a checked-out piece of source material. Files are given a Version when they are checked-out for update so that the changes made during that session can be tracked.